开放搜索

用户指南

用户指南

应用类型

应用类型说明

目前系统支持两种应用类型：标准版与高级版。后续仍会有更多类型推出，请各位关注。

标准版

主要应用在较简单业务场景下，尤其是对数据更新时效性要求高的场景，比如日志、物流、订单、crm等。

主要特征：更新速度快且稳定、仅支持单表

注意

- 标准版应用在配置 RDS 数据源后，就无法再使用SDK API 推送增量数据。即RDS 和 SDK API 只能选择其中一种作为数据推送方式。
- 外网区域，标准版应用中如果配置RDS数据源，则数据源中的过滤条件配置暂不支持。

高级版

主要分为“老高级版”和“新高级版”，主要应用在业务逻辑复杂，或者对搜索效果要求高的场景，比如电商、网页检索、小说、资讯、O2O等。

主要特征：支持简单多表join逻辑、下拉提示、智能分析（同义词、纠错等）。

注意

老高级版应用，目前存在以下2个缺点。如果您对以下2点有要求，强烈建议使用新高级版，新高级版已做过优化。

- 索引重建耗时较长。
- 附表数据生效耗时较长（附表数据不保证生效时间）。

二者功能支持及区别

费用说明详见购买指导部分。

功能列表	标准版	老高级版
多表left join	单表	多表
数据操作命令	支持ADD/DELETE	支持ADD/UPDATE/DELETE
数据处理插件	丰富	丰富
清理过期文档	不支持	支持
清空数据	不支持	支持
RDS自动同步	（华东1/华东2/华北2）支持	（华东1/华北1/华北2）支持
TDDL(mysql)自动同步	支持（仅限内网区域）	支持（仅限内网区域）
ODPS全量自动同步	支持	支持
数据更新时效性	99%文档更新1s内完成	主表90%文档10s内更新完成，附表暂不保证
全量索引多版本	2个	1个
全量版本切换	支持	不支持
索引重建继承数据	应用无数据源配置时，重建生成的新应用版本，不支持继承老应用版本数据，需自行调用SDK推送全量数据。	基于原版本重建，应用无数据源配置时，重建后应用数据不会丢失。
复杂分词支持	丰富	丰富
复杂查询语法	丰富	丰富
统计功能	支持	支持
排序算法	丰富	丰富
LBS服务	支持	支持
结果摘要	支持	支持
查询分析	不支持	支持
下拉提示	不支持	支持

新高级版

因老高级版索引重建耗时无法预估，有时索引重建耗时会比较长，且附表数据生效时间相对也比较长。基于已遇到的问题，我们已对外提供做过性能优化的新高级版。

主要性能优化

- 索引重建速度比老高级版快
- 附表数据生效速度比老高级版快（**新高级版附表数据还是不保证生效时间**）。

新老高级版差异

- 参考官方 “新老高级版差异” 文档。

新老高级版差异

应用相关差异

名称	老高级版	新高级版
应用版本	单应用版本	多应用版本（最多2个版本），支持版本切换服务。

功能相关差异

名称	老高级版	新高级版
下拉提示	支持	支持
清空数据	支持	不支持。
应用迁移	支持（用户在控制台，手动单击应用迁移生成的新高级版，不会继承老高级版中原有的“定时索引重建”和“自动清理过期文档”任务配置。用户需在新高级版应用中重新配置添加。我们协助为用户做“应用迁移”生成的新高级版应用中，已补上这2个配置。）	不支持。

插件相关差异

名称	老高级版	新高级版
应用表字段插件	支持HTMLTagRemover、PinyinConverter	支持HTMLTagRemover，PinyinConverter

注意：老高级版一键迁移的新高级版应用上述插件均支持，新建的新高级版应用中HTMLTagRemover插件改为数据源处配置（不是应用结构），PinyinConverter不再支持，可以使用拼音分词。

API 相关差异

名称	老高级版	新高级版
API 版本	支持V2 和 V3版API	支持V2 和 V3版API**（V2版API仅限查询和推送，不支持应用管控操作，例如不支持查看应用信息）**
API 推送限制	5次/每秒，编码前2M/每次，单条文档大小1M（通常需打包推送）	500次/每秒，编码前2M/每秒，单条文档大小1M（建议打包推送）
API 文档操作命令	支持 ADD、UPDATE**（应用表中不存在对应主键值，转ADD操作）**、DELETE	支持 ADD、UPDATE（**应用表中不存在对应主键值，不更新也不转ADD操作，控制台错误日志中报错。如果是通过应用迁移产生的新高级版，则支持应用表中不存在对应主键值，转ADD操作）**、DELETE
字段忽略（API推送）	推送数据 fields 节点下有包含应用表中不存在字段数据，支持忽略该字段数据进行同步。	推送数据 fields 节点下有包含应用表中不存在字段数据，如果该字段数据格式不符合我们规定语法格式会报错。如果符合我们规定语法格式支持忽略该字段数据进行同步。
系统字段限制（API推送）	推送数据中在与 fields 同级节点下，允许包含不符合我们规定的其它非系统字段数据。	推送数据中在与 fields 同级节点下，只能包含（fields、cmd、timestamp）这3个系统字段数据。如果包含其它非系统字段数据会报错。

字段值相关差异

名称	老高级版	新高级版
INT_ARRAY 字段	如果值为空，默认补 0	如果值为空，默认为空
浮点型字段	保留原有精度	转换为 java double 类型，大于 10^7 和 小于 0.1^4 次方，精度转换成科学计数。例如：5600000000 转换为 5.6E9

索引重建相关差异

名称	老高级版	新高级版
触发带数据源索引重建	触发后立即执行	**通过修改应用结构生成新版本**

		，再单击全量索引构建，触发全量数据导入。否则一直处于等待全量索引构建状态

应用文档大小统计差异

新老高级版，文档大小统计上有所区别。若需将老高级版应用迁移升级到新高级版应用中，且应用中存在多表。后续需重新进行应用容量预估，否则可能报应用容量超配额错误。

注意

华东1区、华北2区：

- 基于兼容性考虑，在 2018-04-20 号之前创建的新高级版，应用容量按主附表 join 后大小计算。
- 在 2018-04-20 号之后生成的新高级版应用（包括应用迁移和索引重建生成的新高级版）和老高级版应用容量计算方式相同（若存在应用容量超配额报错，请先扩容）。

华东2区、华北1区，华南1区：

- 以上3个区域新高级版应用容量计算，临时参考下面按主附表 join 后大小计算。后续上线升级后，新生成的新高级版应用容量计算方式和老高级版相同。

新老高级容量统计示例

- 主表main中有doc1，大小为 2k
- 主表main中有doc2，大小为 3k
- 辅表sub中有doc3，大小为4k，可以分别 join 到 doc1 和 doc2 上

老高级版

- 所有主表文档的总大小 + 所有辅表文档的总大小
- 具体计算公式参考：doc1 + doc2 + doc3 =总doc大小
- 以上示例，总doc大小为： 2 + 3 + 4 = 9k

新高级版

- 累加统计各关联表经过join之后，输出到引擎的文档大小的总和
- 具体计算公式参考：(doc3 + doc1) + (doc3 + doc2)=总doc大小
- 以上示例总文档大小为：(4 + 2)+(4 + 3)=13k

字段类型 和 分词类型

字段类型说明

数据推送到OpenSearch后会先保存到离线数据表中，在此阶段，为了方便用户推送数据，数据表允许用户根据实际业务场景定义多个表（需要指定关联字段），并提供了数据处理的插件。数据处理完毕后会join成一张索引表，这种索引表主要定义搜索属性，供引擎构建索引及查询使用。

这里分别介绍下数据表与索引表的字段对应关系。

数据表字段

数据表主要为数据导入时使用，不同的数据处理插件对类型有不同的要求，这里只是初步类型选择，下一步将有更细化的类型。具体字段取值范围，请参见**系统限制**-字段相关部分说明。超过取值范围将溢出或者截断，请务必保证选择类型正确。

类型	说明
INT	int64整型
INT_ARRAY	int64整型数组
FLOAT	浮点型
FLOAT_ARRAY	浮点型数组
DOUBLE	浮点型
DOUBLE_ARRAY	浮点型数组
LITERAL	字符串常量，仅支持精确匹配
LITERAL_ARRAY	字符串常量数组，单个元素仅支持精确匹配
SHORT_TEXT	短文本，长度在100字节内，支持若干分词方式
TEXT	长文本，支持若干分词方式

支持创建为索引的字段类型

INT，INT_ARRAY，TEXT，SHORT_TEXT，LITERAL，LITERAL_ARRAY

不支持创建为索引的字段类型

FLOAT，FLOAT_ARRAY，DOUBLE，DOUBLE_ARRAY

索引表字段

对于INT及FLOAT类型介绍这里不再累赘（限制详见**系统限制**），重点介绍下各字段类型。

主要类型介绍

搜索效果如何跟分词有很大的关系，分词方式直接影响最终的搜索效果展示，目前系统支持若干的分词方式，需要根据实际业务场景的需求选择合适的字段类型。

接下来，我们详细说明下各个字段的展现效果及适用场景，供大家参考。

不分词

不分词，适合一些需要精确匹配或者只展示不搜索的场景。如标签、关键词、url等，不分词的字符串或数值内容。可以考虑选择不分词的 LITERAL、INT 字段类型。

如文档字段内容为“菊花茶”，则只有搜索“菊花茶”的情况下可以召回。

中文基础分词

按照检索单元做分词，适合有语义的中文搜索场景，如标题、文本等。TEXT及SHORT_TEXT类型可选。

如文档字段内容为“菊花茶”，则搜索“菊花茶”、“菊花”、“茶”、“花茶”等情况下可以召回。

中文单字分词

按照单字/单词分词，适合非语义的中文搜索场景，如小说作者名称、店铺名等。TEXT及SHORT_TEXT类型可选。

如文档字段内容为“菊花茶”，则搜索“菊花茶”、“菊花”、“茶”、“花茶”、“菊”、“花”、“菊茶”等情况下可以召回。

模糊分词

仅适用于SHORT_TEXT短文本类型，支持拼音搜索、数字的前后缀搜索（中文不支持前后缀匹配搜索，字母，数字及拼音，这些都支持前后缀匹配）、单字或者单字母搜索。最多支持100个字节字段长度，更多介绍及注意事项参见模糊搜索使用说明

如文档字段内容为“菊花茶”，则搜索“菊花茶”、“菊花”、“茶”、“花茶”、“菊”、“花”、“菊茶”、“ju”、“juhua”、“juhuacha”、“j”、“jh”、“jhc”等情况下可以召回。
如文档字段内容为手机号“13812345678”，则通过“^138”来搜索以“138”开头的手机号，通过“5678$”搜索以“5678”结尾的手机号。
如文档字段内容为“OpenSearch”，则通过单个字母或者组合都可以检索到。

英文去词根分词

适合于英文语义搜索场景，对于分词后的每个英文单词默认会做去词根、单复数转化。TEXT及SHORT_TEXT类型可选。

如文档字段内容为“英文分词器 english analyzer”，则搜索“英文分词器”、“english”、“analyz”、“analyzer”、“analyzers”、“analyze”、“analyzed”、“analyzing”等情况下可以召回。
（注意：英文分词器中连续的中文会被分成一个词）

英文不去词根分词

适合于英文书名、人名等搜索场景，按照空格及标点符号做分词。TEXT及SHORT_TEXT类型可选。

如文档字段内容为“英文分词器 english analyzer”，则搜索“英文分词器”、“english”、“analyzer”等情况下可以召回。。
（注意：英文分词器中连续的中文会被分成一个词）

拼音全拼

仅适用于SHORT_TEXT短文本类型，支持对短文本中的汉字，按照首字母和拼音全拼进行检索。适用于人名、电影名等需要简拼和全拼搜索的场景，而且全拼检索时必须输入汉字的全拼，不能只输部分。

如文档字段内容为“大内密探007”，则搜索“d”、“dn”、“dnm”、“dnmt”、“dnmt007”、“da”、“danei”、“daneimi”、“daneimitan”等都可以召回。搜索“an”、“anei”等无法召回。

拼音简拼

仅适用于SHORT_TEXT短文本类型，支持对短文本中的汉字，按照首字母进行检索。适用于人名、电影名等需要简拼搜索的场景。

如文档字段内容为“大内密探007”，则搜索“d”、“dn”、“dnm”、“dnmt”、“dnmt0”、“dnmt007”、“m”、“mt”、“mt007”、“007”等都可以召回。

通用分词

适用于新高级版/标准版应用，适用于全网通用行业的分词器。该通用分词也是基于中文语义分词，并且比中文基础分词效果好。

注意

- 华东2区，新高级版应用，不支持该通用分词。

- 属于行业分词类型。

电商分词

适用于新高级版/标准版应用，适用于电商行业的分词器。

注意

- 华东2区，新高级版应用，不支持该通用分词。
- 属于行业分词类型。

简单分词

适合特殊场景下系统自带无法解决的搜索场景，可以实现完全用户控制的效果。推送文档及搜索时使用制表符“\t”对字段内容（或查询词）进行分隔，注意二者分词的一致性，否则会导致无法召回文档的情况。TEXT及SHORT_TEXT类型可选。

如字段内容为“菊\t花茶\thao”，则只有查询词“菊”、“花茶”、“菊\t花茶”、“花茶\thao”、“菊\thao”、“菊\t花茶\thao”可以召回该文档。

自定义分词

适用于新高级版/标准版应用，参考自定义分词器文档。

自定义分词器方式

行业分词器 + 干预词典。

分词测试

前往应用控制台的应用列表界面 -> 高级配置 -> 自定义分词器 -> 分词测试。

注意

- 华东2区，新高级版应用对应的自定义分词，是对应上面的简单分词效果。

适用场景

- 有语义环境的中文搜索，建议使用中文语义分词；
- 对于短文本或者非语义环境中文搜索（对排序没有太多要求），建议使用中文单字分词来扩大召回；
- 拼音搜索请使用模糊分词；
- 英文场景下请使用英文去词根分词；
- 某些场景下，中文语义分词及单字分词搭配使用，可以获得非常好的搜索效果。如查询

query=title_index:’菊花茶’ OR sws_title_index:’菊花茶’ ，精排表达式为 :text_relevance(title)*5+field_proximity(sws_title)。可以实现包含“xx菊xx花xx茶xx”的文档，且排序上“菊花茶”会排在前面。

注意事项

- 如果TEXT字段设置了搜索结果摘要，扩展检索单元部分词组（如上例中的“花茶”）将不会被添加飘红标签。
- 中文单字分词对于数字跟单词认为是一个词，如“hello word”，搜索“hello”可以召回，搜索“he”则无法召回，敬请注意。若需要做单词内召回，请选择模糊分词。
- 应用结构中的主表的主键，默认会被设置为索引字段，且索引字段名称默认为“id”，不支持修改配置。

系统限制

系统相关

项	值
每个用户应用数	不限制
每个用户doc总数	理论上不限制，具体根据应用配额文档容量来计算
每个用户pv总数	理论上不限制，具体根据应用配额QPS峰值来计算
系统支持汉字编码	UTF-8

应用相关

项	值
应用名长度	30字符
应用字段名长度	30字符
排序表达式名称长度	30字符
附表个数	10
应用字段个数	256
源表表名长度	16字符
索引字段名	64字符
源表外表关联层级	2级

INT类字段个数	256
LITERAL字段个数（不支持创建为组合索引）	256
TEXT类型字段个数	32
组合索引个数	4个
组合索引包含字段数	8个
TEXT类型单字段索引个数	32个
LITERAL类型单字段索引个数	256个

字段相关

项	值
INT64	-2^63~2^63-1
FLOAT	+/-3.40282e+038
DOUBLE	+/-1.79769e+308
LITERAL	64K个字符
TEXT	64k个词
ARRAY	64K个元素(性能消耗大，100个元素内获得最佳性能）

排序表达式

项	值
粗排表达式条数	30个
精排表达式条数	30个

下拉提示

项	值
每个应用下拉提示规则数目	3
每个下拉提示规则包含字段数	3
每个下拉提示规则包含黑名单条目	500
每个下拉提示规则包含推荐词条条目	500

搜索结果摘要

项	描述	取值范围
片段长度	表示摘要长度	[1-300] 字节
片段数量	在摘要长度内需要几个片段	[1-5]

推送数据【应用级别】（老高级版）

项	值
API 每秒推送次数	5次（通常需打包推送）
每个API 请求包大小	编码前2M
每条文档大小	1M
RDS增量同步速率	1500条记录/秒/实例
TDDL增量同步速率（仅内网支持）	1500条记录/秒/库
增量处理时效性	90%的文档推送成功后可以在10s内搜索到，99%在10min内，附表暂不保证。

推送数据【应用级别】（标准版）

项	值
API 每次推送总文档数	1000个，建议100个性能更好（建议打包推送）
API 每秒推送总次数	500次
API 每次请求总容量	编码前2M
API 每秒请求总容量	编码前2M
RDS增量同步速率	拉取数据，编码前2M/秒/应用。 写入数据，编码前5M/秒/应用。
TDDL增量同步速率（仅内网支持）	拉取数据，编码前2M/秒/应用。 写入数据，编码前5M/秒/应用。
每条文档大小	1M
增量处理时效性	99%的文档推送成功后可以在1s内搜索到，99.9%在1min内

推送数据【应用级别】（新高级版）

项	值
API 每次推送总文档数	1000个，建议100个性能更好（建议打包推送）

API 每秒推送总次数	500次
API 每次请求总容量	编码前2M
API 每秒请求总容量	编码前2M
RDS增量同步速率	拉取数据，编码前2M/秒/应用。 写入数据，编码前5M/秒/应用。
TDDL增量同步速率（仅内网支持）	拉取数据，编码前2M/秒/应用。 写入数据，编码前5M/秒/应用。
每条文档大小	1M
增量处理时效性	90%的文档推送成功后可以在10s内搜索到 ，99%在10min内，附表暂不保证。

推送数据中不能包含下列系统保留不可见字符

编码	(emacs/vi)中的显示形态
“\x1E\n”	^^
“\x1F\n”	^_
“\x1D”	^]
“\x1C”	^\
“\x1D”	^]
“\x03”	^C

搜索相关

项	值
每个子句(除filter)最大长度	编码后1k
filter子句最大长度	编码后4k
请求最多返回结果数	5000
参与粗排文档数	100万
参与精排文档数	默认200
粗排字段	4个

创建应用

创建标准版

内网用户

详情请参见 标准版应用接入流程。

外网用户

创建标准版与创建高级版流程基本一致，主要区别是标准版不支持多表。因此您可以直接参考下面的“创建高级版应用”流程。

创建高级版

填写基本信息

定义应用结构

目前提供了 4 种方式的应用结构创建方式，同时OpenSearch高级版提供了多表支持功能，以方便业务复杂场景下调用。

主辅表数据关联关系

通过手动创建应用结构方式，为应用创建多个表时，多表之间数据关联关系描述如下：

- **目前主辅表，仅支持 N:1 或 1:1 的关系，不支持 1:N（即多表数据关联关系中，多的一方只能是主表，且主表只能有1个）。**
- **主辅表需通过应用表外键与附表主键进行数据关联，且表外键只能关联辅表主键。**
- **最多只支持2层关联。**

多表数据关联支持

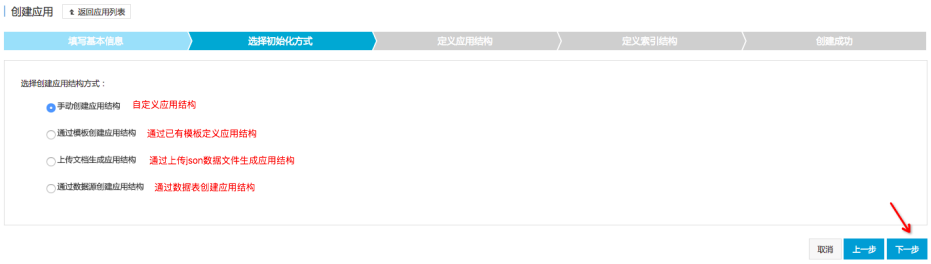
- 表a->表b，表b->表c
- 表a->表d

不支持超过2层多表数据关联

- 表a->表b，表b->表c，表c->表d

不支持环状多表数据关联

- 表a->表b，表b->表a



- 1、**手动创建应用结构。**可以自定义应用结构进行应用创建。
- 2、**通过模板创建应用结构。**系统默认提供了几种常用的模板样式，用户也可以将自己定义的应用结构创建成模板，可以通过已有模板快速创建出一个新的应用。
- 3、**上传文档生成应用结构。**您可以上传已有的数据文件（仅支持JSON格式），系统会自动解析并创建出初始的应用结构（注意字段类型等需要重新定义）
- 4、**通过数据源创建应用结构。**适用于通过RDS、ODPS等数据源同步的场景，可以快速由源表结构创建出初始的应用结构，节省手动构造的工作量，降低出错概率。这里以RDS为例，其他数据源操作类似，具体详见数据源配置

- 4，**通过手动方式创建应用结构**。非以上三种场景使用。

创建索引和属性

- 需放到 query子句中的字段，必须创建为索引（浮点型不支持创建为索引）
- 需放到 filter子句，sort子句，及函数中涉及字段有明确标识，需设置为属性的字段必须创建为属性。

注意

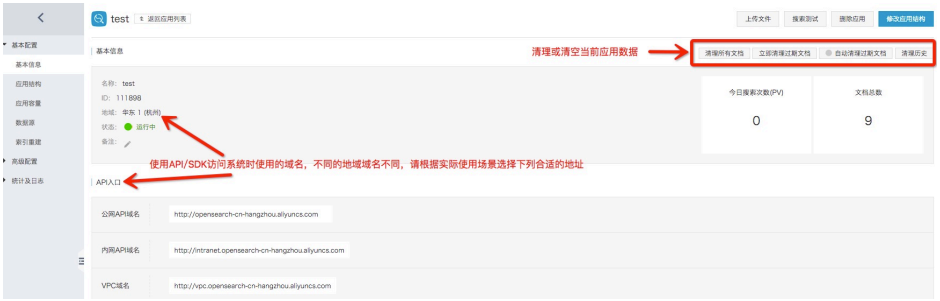
- 分词字段类型无法配置为属性，例如 TEXT，SHORT_TEXT等都不支持，只支持数值字段类型及不分词字段类型配置为属性，例如
  int，int_array，float，float_array，double，double_array，literal，literal_array 等字段类型。

创建应用并激活

应用创建及数据源配置

基本信息

该部分主要罗列了应用的基本信息，以及API入口、访问数据指标、数据清理功能配置。



基本信息
包含应用的名称、id、所在地域、运行状态、备注等。

API入口
每个地域的API域名不同，使用API或者SDK访问OpenSearch的时候需要根据此处给出的API入口进行访问。同时不同地域也根据使用的场景区分了内网（ECS可用）、公网、VPC域名，请根据实际部

署情况选择合适的域名，使用前请先ping下，确定可访问。

数据清理
可以进行数据清空（数据量较大的情况下建议直接删除应用重新创建的方式效率更高）、自动（立即）请求过期数据，所有的清理记录都可以在清理历史中查询。

应用容量

配额管理

配额管理是开放搜索提供的一种对用户使用资源进行有效管理的功能。对不同应用进行资源协调，防止应用间相互干扰，从而为用户提供更稳定的服务。

目前配额主要包含两部分：LCU及存储容量，可以让用户按需使用。目前已开放预警功能，报警信息会发送到用户的邮箱中，请及时调整，避免出现超过配额被拒绝的情况。

LCU及应用容量配额修改

- LCU及应用容量在非人工审批范围内，可随时修改调整
- 支持LCU及应用容量缩容，应用容量缩容范围不可低于当前已使用容量大小
- 应用容量在超配额报错且经过后续扩容之后，之前丢失的数据不会自动追加，需要重新导入数据索引重建

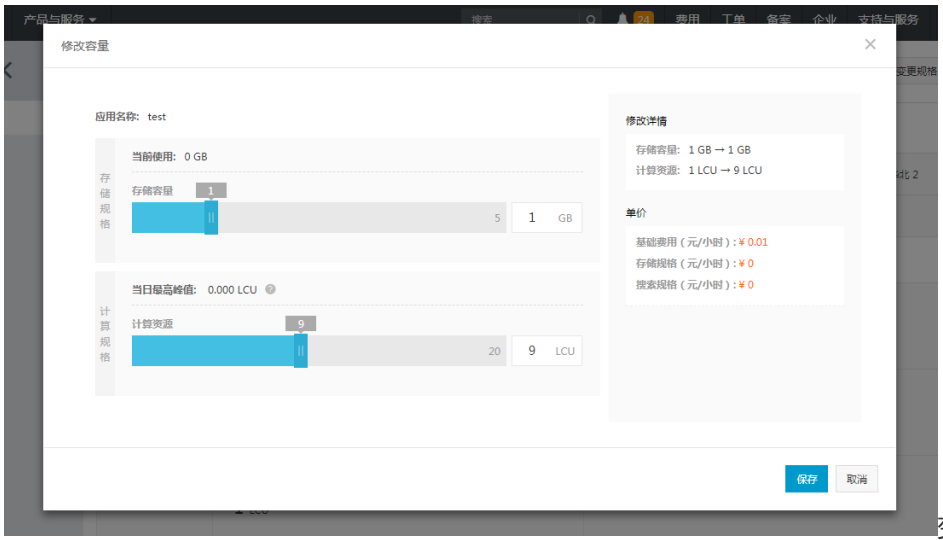
流程演示

1.主界面

2.扩容

点击应用名——基本信息——右上角“扩容”；或者应用列表右侧“扩容”

3.变更实例以及修改配额



仅修改配额：变更实例及修改配额：（变更规格时，需要调整具体的存储及LCU的值到高于原来价格，才能够点击“保存”；变更规格后

，可以再缩容。 ）

4.审批状态：专享型集群需要审批，其他规格不需要审批，即时生效。

数据源及插件简介

OpenSearch中的数据，既支持通过API/SDK/上传界面的方式导入，也支持直接从已有的云端数据源进行同步。如果选择通过API或SDK来上传数据，可以参照API手册直接上传，无需额外配置。如果选择同步云端数据的方式，则需要将数据源的相关信息在控制台中进行配置。目前系统提供了若干的数据处理插件可以实现一些简单的数据转化操作，在配置数据源字段对应关系（API方式上传数据的暂不支持，需要用户推送前处理好）可以选择使用。

一张OpenSearch表可以支持多个rds及TDDL(mysql)来源表（如分库分表的场景），但是ODPS源只能配置一个，如需多个ODPS来源表，请先将数据合并成一张表后再导入。

数据处理插件

系统中某些搜索功能或者特征函数需要特殊的字段类型支持。如Array类型字段，需要通过如下插件来转化，用户无法直接输入。

注意：该插件在数据源配置处配置，而不是定义应用结构的时候

配置项名称	说明	示例	版本
JsonKeyValueExtractor	从Json格式的来源字段中提取指定的键值，提取出来的键值作为目标表字段的内容，只能抽取某个key中的值。	{ “title” :” the content” ,” body” : ” the content” } 中提取出键值title的内容，若内容为JsonArray格式，则将转化为系统中Array类型字段内容 “请确保提	高级版/标准版

		取出来的键值和目标表字段类型一致，否则对应的数据会丢失”。此处的JsonArray格式，是指符合我们这边定义的JsonArray格式。例如 literal_array字段类型：{“tags”:[“a”,”b”,”c”]} 或 int_array字段类型：{“tags”:[1,2,3]}	
MultiValueSpliter	将来源字段按照分隔符分割成多个值，分割后的内容作为目标表字段的内容，目标表字段必须是配置为ARRAY类型的字段 (若分隔符为不可见字符，需要使用unicode字符来标识，如\u001D)	数据源内容为：1,2,3，指定分隔符为 “,” 直接输入一个英文的逗号即可	高级版/标准版
KeyValueExtractor	从KV格式的来源字段中提取指定的键值，提取出来的键值作为目标表字段的内容，只能抽取某个key中的值。分隔符可以不填	实际内容为：key1:value1,value2;key2:value3，键为key1,key2，键分隔符为分号，键值分隔符为冒号，多值分隔符为逗号。如果配置了多值分隔符，则将转化为系统中Array类型字段内容“请确保提取出来的键值和目标表字段类型一致，否则对应的数据会丢失”，若存在2个相同的key，则只会抽取后面的那个key的值。	高级版/标准版
StringCatenateExtractor	将多个指定字段按照指定的顺序拼接成一个字符串，该插件不支持int字段类型，建议用literal字段类型；字段列表以逗号分隔（字段需来自于目标字段）	将field1，field2内容按照‘_’组成新的字段内容。另系统变量$table可以获取当前表名	高级版/标准版

API/SDK数据源

通过API/SDK导入非常灵活，完全由用户控制，具体请参见API开发者指南及Java SDK文档及Php SDK文档。

TDDL(Mysql)数据源配置

仅“内网杭州”区域可见，请移步TDDL对接OpenSearch流程。

索引重建

对于用户上传的数据（包括通过各个数据源的同步过来的数据）OpenSearch会在系统中保存一份镜像。如果有涉及到应用结构变更、或者需要导入全量数据的情况下，需要进行索引重建操作。目前支持两种索引重建方式：1）手动索引重建（一般用于修改应用结构或者导入全量用户数据时使用）；2）每日定时任务（一般在odps等数据源每天导入全量用户数据使用。RDS默认开启数据同步，无需配置定时任务）。

定时任务与手动任务的逻辑完全相同，只需要多配置一个每日同步的时间。**需要注意的是：定时任务每天只会执行一次，一旦当天成功执行了一次，无论如何修改配置，都不会再次执行。**

在索引重建任务开始之前，需要选择任务的类型：

- 只重建索引：对应于应用结构有变化的情况，如果选择了这个操作，仅仅会重新构建应用的全部索引，不会拉取数据源中的数据
- 重新导入数据并重建索引：一般对应于首次向OpenSearch中导入全量数据的场景，或任何需要从数据源中拉取全部数据，并重建索引的场景。在“重新导入数据并重建索引”的任务中，可以选择一张

或多张表进行同步（也就是说不必须是全部的表）。OpenSearch会根据所选择的各个表之间的关系自动确定导入顺序。

任务成功创建之后，会显示任务执行的进度，点击进度条，可以查看进度详情。如果任务失败，可以在应用列表页中的错误日志中查询失败原因。

索引重建流程及进度

流程为：数据导入——数据处理——索引构建。

从数据源导入数据后，做数据处理，发给build服务做索引构建，全量导入的build任务是定期执行一轮，一轮的时间是30分钟左右。因此文档数很少的情况下，也可能需要一些时间完成全量任务。控制台的索引重建进度为阶段性展示。

注意：

新高级版以及标准版，使用“手动索引重建”或者“修改应用结构”生成一个新版本后，一个应用实例会在控制台存在新老两个版本。

- 当存在两个版本时，只有一个版本的状态为正常（服务中），表示该版本当前提供搜索服务（当使用appname调用search接口时，默认搜索的版本）。
- 当对某个版本执行了“切换到线上”的操作时，意味着另外一个版本被切换下线。该被切换下线的版本默认保留8小时，8小时后，该版本会被自动删除。
- 版本删除后，不可恢复。

清理过期文档

开放搜索已提供“清理过期文档”功能。可以通过在控制台实现，手动清空应用文档、手动清理过期文档、自动清理过期文档、以及查看历史任务等需求。

您可以登陆OpenSearch控制台，单击应用“管理”进入应用基本信息界面，在页面右上角分别有“清空数据”、“清理过期文档”、“历史任务”三个功能按钮。

注意

- 清空数据、清理过期文档等功能，都是走索引重建方式（会计费）。
- 清空数据功能，是将整个应用中的文档清空。

- 清理过期文档下面的功能，只能清理主表中符合条件的过期文档。

支持应用类型

老高级版

主要支持“清空数据”、“清理过期文档”、“历史任务”这3个功能按钮。

新高级版

主要支持“清空数据”、“清理过期文档”这2个功能按钮。

标准版

不支持清理过期文档功能。

清空数据

单击清空数据按钮，出现如下界面按提示输入应用名，再单击清空按钮执行清空任务。

清空数据任务运行时，如下图所示。

清理过期文档

清理过期文档功能按钮下面，可以再选择“立即清理”和“自动清理”。

立即清理

- **保留字段**：必须为主表中的 INT 字段类型，该字段值为“秒级”或“毫秒级” unix 时间戳值。
- **在最近X天内的文档**：这里的 X 表示需保留最近多少天内的文档，值域 [7-180]。
- **该字段值单位为**：
  - 秒： 秒级 unix 时间戳一般为10位整数。
  - 毫秒： 毫秒级 unix 时间戳一般为13位整数（数据源中datetime类型，OpenSearch将转为以毫秒为单位）。

注意

- 保留字段只能是主表中的 INT 字段类型。

自动清理

- **每天自动清理文档**：表示是否开启自动清理过期文档功能（默认关闭）。
- **保留字段**：必须为主表中的 INT 字段类型，该字段值为“秒级”或“毫秒级” unix 时间戳值。
- **在最近X天内的文档**：这里的 X 表示需保留最近多少天内的文档，值域 [7-180]。
- **该字段值单位为**：

- 秒： 秒级 unix 时间戳一般为10位整数。
- 毫秒： 毫秒级 unix 时间戳一般为13位整数（数据源中datetime类型，OpenSearch将转为以毫秒为单位）。

注意

自动过期清理将每天清除您选择的字段中符合过期条件的文档，更新频率较低时请不要使用该功能。

历史任务

可查看历史清理文档任务相关信息。

ODPS数据源配置

开放数据处理服务（Open Data Processing Service， 简称ODPS）是一个开放的计算平台，如果您要导入到OpenSearch的数据是由ODPS平台计算而产生的，则可以在应用中配置ODPS源信息，在触发应用索引重建任

务后，系统会自动去获取 ODPS 表中的全量数据，后续的增量需通过调用SDK API推送过来。

目前 ODPS 数据源只支持全量同步，不支持增量同步。

【需注意】ODPS内外网分离，即外网ODPS在内网区域使用会有问题，所以在使用上有很多注意事项，我们整理了接入流程，请移步OpenSearch对接ODPS（云梯2）流程。

1. 入口有两个:

在应用基本配置-数据源中选择ODPS作为数据来源；

或者创建应用的时候直接配置ODPS源。详见通过ODPS创建应用。

2. 配置ODPS源信息

OpenSearch支持当前账号下的ODPS的project，或者已经授权给当前账号访问的project中获取数据。选择“ODPS”数据源后，选择“被授权的project”，输入odps中要访问的project信息进行连接校验（已成功连接的project系统会进行缓存，直接点击对应的project名称即可，无需重新连接）。

如果连接校验失败，则需要检查授权是否存在或授权最近有无变更过。（需注意ODPS表字段若没有权限或权

限不对，也会报错。）

配置字段映射关系：OpenSearch为ODPS源的数据提供了若干数据转换插件，如要使用，则在配置字段对应关系的同时，点击“内容转换”列中的“+”符号，则会在源字段被同步到OpenSearch之前，先进行内容转换

，再进行同步。

如果内容转换插件由于配置错误、无法连接等错误失效，则源字段仍然会被同步到目标字段，只是内容不会被转换。

【注意】对于ODPS表中的datetime及timestamp类型系统会自动转化为毫秒数，请将对应OpenSearch字段类型设置为INT。

3 .选择分区信息

3.1 根据ODPS数据特性，OpenSearch允许用户根据具体需要来指定导入的分区，高级版支持正则表达式，表示导入前一天的数据，结合应用基本信息-索引重建-定时索引重建功能，可以实现每天导入新分区数据的效果。

3.2 **标准版以及高级版均支持正则表达式** (等号/逗号/分号/双竖线为系统保留字符，分区列名/列值中应避免出现这些字符):

【高级版/标准版应用每天自动导入前1天分区全量数据条件例子】 pt=%Y%m%d || -1 days 【注：pt为分区字段名】

不同场景下odps分区条件用法，参考如下所示：

1: 支持多个分区过滤规则，不同的分区过滤规则用分号分隔，如pt=1;pt=2将匹配满足分区字段pt=1或者pt=2的所有字段

2: 分区过滤规则，支持指定多个分区字段的值，不同分区字段用逗号分隔，如：pt1=1,pt2=2,pt3=3将匹配同时满足pt1=1，pt2=2，pt3=3的所有分区【分区中若存在多个字段，则多个字段都必须要指定，否者会报错】

3: 分区字段的值支持通配符 *，表示该分区字段可以为任意的值，这种情况下，过滤规则中也可不写

该字段

4: 分区字段的值支持正则表达式，如pt=[0-9]* 将匹配pt值为数字的所有分区

5: 分区字段的值支持时间匹配，匹配规则： pt=包含格式化时间的分区列值||时间间隔表达式。如ds=%Y%m%d || -1 days，表示分区字段为ds，格式为20150510，需要访问1天前的数据。

5.1 格式化时间参数支持标准的时间格式参数，如下表

5.2 时间间隔表达式支持 +/- n week|weeks|day|days|hour|hours|minute|minutes|second|seconds|microsecond|microseconds，+号任务创建时间的表示n周/天/小时/分钟/秒/毫秒后，-号表示任务创建时间的表示n周/天/小时/分钟/秒/毫秒前。

5.3 系统默认会对所有过滤规则，按照+0 days进行时间参数替换，因此，需要注意的是，用于过滤的字段值不能包含下面这些字符串作为普通的字符串参数，如星期三创建的任务，pt=%abc 将匹配pt的值为Wedbc的分区，而不是pt=%abc的分区。

正则表达式全部可用参数及含义，参考如下：

```
%a: 星期的简写。如 星期三为Wed
%A: 星期的全写。如 星期三为Wednesday
%b: 月份的简写。如4月份为Apr
%B: 月份的全写。如4月份为April
%c: 日期时间的字符串表示。（如： 04/07/10 10:43:39）
%d: 日在这个月中的天数（是这个月的第几天）
%f: 微秒（范围[0,999999]）
%H: 小时（24小时制，[0, 23]）
%I: 小时（12小时制，[0, 11]）
%j: 日在年中的天数 [001,366]（是当年的第几天）
%m: 月份（[01,12]）
%M: 分钟（[00,59]）
%p: AM或者PM
%S: 秒（范围为[00,61]，为什么不是[00, 59]，参考python手册~_~）
%U: 周在当年的周数当年的第几周），星期天作为周的第一天
%w: 今天在这周的天数，范围为[0, 6]，6表示星期天
%W: 周在当年的周数（是当年的第几周），星期一作为周的第一天
%x: 日期字符串（如：04/07/10）
%X: 时间字符串（如：10:43:39）
%y: 2个数字表示的年份
%Y: 4个数字表示的年份
%z: 与utc时间的间隔 （如果是本地时间，返回空字符串）。
```

4 . 选择数据同步并发控制机制(目前仅“内网杭州”区域可见)

当用户勾选【使用done文件】后，OpenSearch支持用户通过上传done文件的方式控制系统拉取全量数据的时机，保证全量数据的完整性。系统在开始从odps拉全量数据之前会先判断一下当天的done文件是否存在

，如果不存在则等待，默认等待1小时后超时。

- 用户需从odps官网下载odps clt安装包；
- 用户需要具有所在project空间的CreateResource权限；
- 安装后在用户程序中运行如下命令：其中done文件的命名规则为$prefix_%Y-%m-%d。$prefix: 文件名前缀，默认为表名，%Y-%m-%d：索引重建任务日期，系统定时任务目前支持的最小粒度为1天。

```
odpscmd –u accessid –p accesskey --project=<prj_name> –e "add file <done file>;"
```

done文件内容为json格式，目前仅需包含如下内容，用于指定该批全量数据的时间戳（毫秒）【最多只保留3天增量，因此该时间点不可以超过3天】。

该时间戳表示需要回溯的增量数据时间点，如果不配置则默认从索引重建任务开始时间追加数据【最多只保留3天增量，因此该时间点不可以超过3天】。

【例如】全量数据是今天9点的，odps处理完毕后为10点，OpenSearch定时任务为10:30，则done文件需要指定为当天9点的毫秒值，在处理完全量后系统会追加当天9点后的增量，保证数据完整性；否则会从默认任务启动时间10:30开始追加，这样9:00~10:30期间的增量会丢失，该行为非常重要，需要特别注意。（当然，若没有增量，则无需配置该时间戳）

高级版done文件内容如下所示（提示：标准版中需设置的数据时间值也是类似原理，都是用来追期间增量的）

```
{
"timestamp":"1234567890000"
}
```

RDS数据源配置

云数据库（Relational Database Service，即关系型数据库服务，简称RDS）是阿里云对外提供的一种即开即用、稳定可靠、可弹性伸缩的在线数据库服务（了解RDS ）。

购买RDS前须知

- 购买RDS实例时，建议选择5.6版，并且必须是常规实例，双机高可用版（不支持5.2以下，5.7及以上版本）。

- RDS实例必须隶属于当前登录阿里云账号才能访问使用。
- RDS实例所在区域必须与OpenSearch应用区域一致。
- 不支持**RDS clone**实例，否则应用激活后状态一直处于初始化。

温馨提示：

- 外网区域：华东1、华东2、华北1、华北2、华南1 等，不支持DRDS数据源。

支持功能

- 支持增量实时同步（默认勾选）。
- 支持（手动/定时）拉取指定数据库表全量。
- 支持单个或多个数据源表数据横向合并，要求这些源表结构及数据源插件配置必须完全相同，并且主键值均不重复（主键值重复会覆盖），主要支持以下2种场景：
    - 应用表中配有一个数据源，并且包含多个源表。
    - 应用表中配有多个数据源，并且各数据源包含1个或多个源表。
- 支持数据源字段转换插件。
- 新老高级版的RDS数据源，支持（全量/增量）过滤条件。
- 支持通过通配符*匹配数据库表名。

相关限制

- 只支持RDS的binlog为**full**模式（否则拉取增量会不全）。
- 只支持RDS中的Mysql常规实例 （双机高可用版）。
- 不支持5.2以下，5.7及以上的RDS版本。
- RDS实例必须隶属于当前登录阿里云账号才能访问。
- RDS实例所在区域必须与OpenSearch应用区域一致。
- 标准版应用在配置RDS数据源后，不支持（SDK/API）推送增量。
- （外网区域）标准版应用的RDS数据源，暂不支持数据源过滤条件。
- 不支持**replace into**语法。
- 不支持**truncate**和**drop**命令，请使用**delete**命令删除数据。
- 不支持通过**视图**同步增量。
- 不支持**RDS clone**实例，否则应用激活后状态一直处于初始化。
- RDS 访问密码不能包含%符号，会导致索引重建任务失败。
- 不支持通过RDS**高权限账号**访问。（否则连接RDS会失败）。
- 不支持在不同数据库源表结构之间做字段列合并。

注意事项

- RDS支持内/外网的域名切换，OpenSearch对RDS数据获取均不收取任何流量费用。
- OpenSearch仅支持从主库拉取全量数据，建议根据您的业务繁忙情况，选择低峰期索引重建导入全量数据。
- RDS表中**datetime**及**timestamp**此类时间类型，系统会自动转化为毫秒数，请将对应应用表字段类型

设置为INT。
- 外网区域RDS实例，需要申请内网地址后才能访问，否则会提示**连接RDS服务失败，请稍后再试。**
- 不符合数据源过滤条件的（增量/全量）文档会被过滤，**并且如果对应应用表中存在相同主键值的文档也会一并删除。**

常见问题

- 使用RAM子账号在控制台中为应用配置RDS数据源，必须要对该RAM子账号进行授权，否者会提示**连接RDS服务失败，请稍后再试**，参考 **授权访问鉴权规则** 文档。
- 如果老高级版应用RDS实例期间欠过费，但后续有将欠费补上，您需要向我们提工单反馈做下处理，否则后续增量将无法同步（如果是标准版/新高级版应用，您可以通过触发一次索引重建，新应用版本可以正常同步增量）。
- RDS访问密码不能包含%符号，否则会导致索引重建任务失败（报错提示： Illegal hex characters in escape (%) pattern）。
- 系统要求应用表主键值唯一，如果分表情况下主键值有重复会覆盖，可使用StringCatenateExtractor数据源插件合并多个字段值，来源字段为pk,$table（pk替换为RDS表主键字段，$table为默认系统变量，表示对应数据库表名），拼接字符为-（可自定义）。
  例如，RDS表为my_table_0，主键字段值为123456，拼接后新主键值为123456-my_table_0。
- 根据数据库表中的**date**或**datetime**字段类型过滤数据，假设数据库表字段名为createtime，则数据源过滤条件中的时间格式必须为 createtime>'2018-03-01 00:00:00'，如果使用 createtime>'2018-3-1 00:00:00'这种格式会报错。

配置RDS

- 在创建应用过程，配置RDS数据源。
- 已创建应用可通过应用数据源界面修改，或通过修改应用结构流程进行修改。

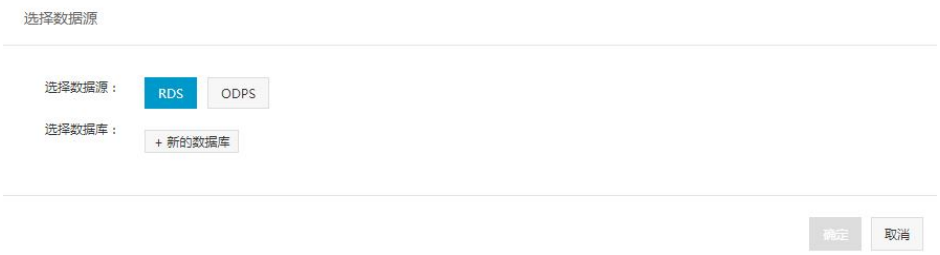
步骤如下

此处以创建应用流程，配置RDS数据源为例。

1 . 在创建应用时，选择需要的应用版本类型，并填写应用名。

2．在第2步中选择**通过数据源创建应用结构。**

3．选择RDS数据源，点击**新的数据库。**



4．RDS信息填写完成后，点击连接按钮。

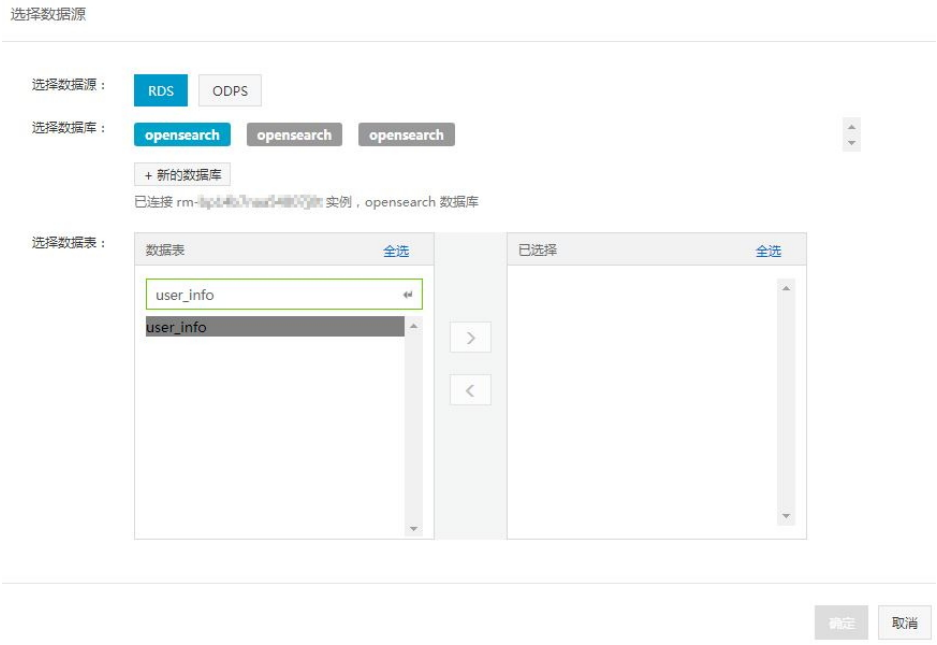
参数名称	说明
实例ID	RDS数据库的实例ID（不是名称），可以在RDS控制台中获取（大小写敏感），暂不支持只读实例，需填写的实例ID格式参考： rm-bp19b4g5n11111111
数据库名	该实例下需要连接的数据库名（大小写不敏感）。
用户名	数据库只读账号，用于获取数据库表模式及全量数据（大小写敏感且具有只读权限）。
密码	只读帐户对应的密码。
授权	对OpenSearch获取RDS数据做授权操作，在用户设置了IP白名单的情况下，OpenSearch会将所用

	IP添加到用户白名单中。若不勾选授权，则**连接数据**不会高亮显示。

OpenSearch会尝试连接，并根据具体情形，给出结果提示：

提示信息	处理方法
当前用户的当前区域没有此RDS实例	请检查实例ID是否正确，并确保RDS实例所在区域与OpenSearch应用区域一致。如果条件符合仍然报错，可提工单反馈
连接RDS服务失败	请检查RDS连接串是否正确包括实例ID、数据库名、用户名、密码
当前RDS数据库下没有此表	请检查表名填写是否正确,以及RDS数据库中是否确实存在该表

5 . 已建立数据源连接界面如下，选择对应表，并单击向右箭头，保存到右侧已选择界面。



- 选择或输入该数据库下需要访问的表名（大小写敏感）。
- 支持分表规则 table_* 的方式，例如 table_a、table_b 等。

6 . 若连接成功，则进行字段配置，OpenSearch会自动获取表字段。

7 . 定义索引和属性等信息。

8 . 配置RDS数据源过滤条件。

- OpenSearch应用表中也可配置多个数据源，但最终这些表结构及配置必须完全相同。
- OpenSearch的全量数据过滤方式为，将过滤条件直接增加在SQL语句的where条件中。 如果应用无需使用增量数据，过滤条件数值部分可以替换为表达式，与数据库中支持的表达式一致。

参数名称	说明
过滤条件	需填写数据库表字段，该过滤条件会同时作用于全

	量和增量数据（如果开启同步）。 支持如下格式：数据库字段 user_id（<、>、<=、>=、=、!=）数值。 多个过滤条件之间AND关系，必须要使用英文逗号（,）分隔，表示且的关系（暂不支持或关系）。例如当过滤条件为 user_id=1,level=1 时，则只能拉取符合该条件的记录。 同步过来的增量文档，若不符合过滤条件，会删除应用中已存在的对应文档。 如果需要根据db表中的date或datetime字段进行过滤，假设db字段名为createtime，则数据源过滤条件中的时间格式必须需为 createtime>'2018-03-01 00:00:00'
数据自动同步	是否自动同步用户数据库的增量数据（默认开启）。

9 . 编辑数据源，映射需要拉取的数据库字段。单击保存，完成应用创建。

- 在该界面可以添加需要映射同步的数据库字段。
- 在该界面中的内容转换，可以添加数据源插件。

10 . 单击激活应用，并选择合适的存储容量进行激活。若是新高级版和标准版，需再单击控制台中的全量索引构建才能运行。

应用高级配置

搜索相关性配置

排序表达式（Ranking Formula）允许用户为应用自定义搜索结果排序方式，通过在查询请求中指定表达式来对结果排序。排序表达式支持基本运算（算术运算、关系运算、逻辑运算、位运算、条件运算）、数学函数和排序特征（feature）等。Open Search对于几种经典的应用（如论坛、资讯等）提供了表达式模板，用户可根据自己数据的特点，选择合适的表达式模板，并以此为基础进行修改，生成自己的表达式。

在进行相关性排序之前，首先要了解下系统排序策略：通过query等子句找到符合条件的文档后，会进入排序阶段（具体参见sort子句），如果未指定sort子句或者sort子句中显式指定了RANK，那么都将进入到相关性算分阶段。

搜索引擎对于检索性能要求比较高，为此，系统开放了两阶段排序过程：粗排和精排。粗排即是海选，从检索结果中快速找到质量高的文档，取出TOP N个结果再按照精排进行精细算分，最终返回最优的结果给用户。由此可见，粗排对性能影响比较大，精排对最终排序效果影响比较大。因此，粗排要求尽量简单有效，只提取精排中的关键因子即可。

如何设计粗精排公式要取决于实际搜索场景的需求，最佳实践-功能篇有个《相关性实战》的文章，较详细介绍了在几个典型场景下如何来思考和设计排序因子，大家可以参考。

注意：

粗精排表达式中一律使用 数值或数值字段类型 参与基本运算操作，例如算数，关系，逻辑，条件等运算

操作，大部分函数都不支持字符串类型进行运算。

基本运算

运算	运算符	说明
一元运算	-	负号，功能为对某个表达式的值取负值，如-1, -max(width)。
算数运算	+, -, *, /	如width / 10
关系运算	==,!= ,>, <, >=, <=	如width>=400
逻辑运算	and ,or,!	如width>=400 and height >= 300, !(a > 1 and b < 2)
位运算	&, |,^	如 3 & (price ^ pubtime) + (price | pubtime)
条件运算	if(cond, thenValue, elseValue)	如果cond的值非0，则该if表达式的实际值为thenValue，否则为elseValue。如if(2, 3, 5)的值为3，if(0, 3, 5)的值为5。（注意：不支持字符串字段类型，如literal或text类型都不支持）
in 运算	i in [value1, value2, ..., valuen]	如果i的值在集合[value1, value2, ..., valuen]中出现，则该表达式值为1，否则为0。例如： 2 in [2, 4, 6]的值为1，3 in [2, 4, 6]的值为0。

数学函数

函数	说明
max(a, b)	取a和b的最大值。
min(a, b)	取a和b的最小值。
ln(a)	对a取自然对数。
log2(a)	对a取以2为底的对数。
log10(a)	对a取以10为底的对数。
sin(a)	正弦函数。
cos(a)	余弦函数。
tan(a)	正切函数。
asin(a)	反正弦函数
acos(a)	反余弦函数
atan(a)	反正切函数。

ceil(a)	对a向上取整，如ceil(4.2)为5。
floor(a)	对a向下取整，如floor(4.6)为4。
sqrt(a)	对a开方，如sqrt(4)为2。
pow(a,b)	返回a的b次幂，如pow(2, 3)为8。
now()	返回当前时间，自Epoch (00:00:00 UTC, January 1, 1970)开始计算，单位是秒。
random()	返回[0, 1]间的一个随机值。

内置特征函数

OpenSearch提供了丰富的**内置特征函数**，如LBS类、文本类、时效类等，可以用在排序表达式中，相互组合实现强大的相关性排序效果。

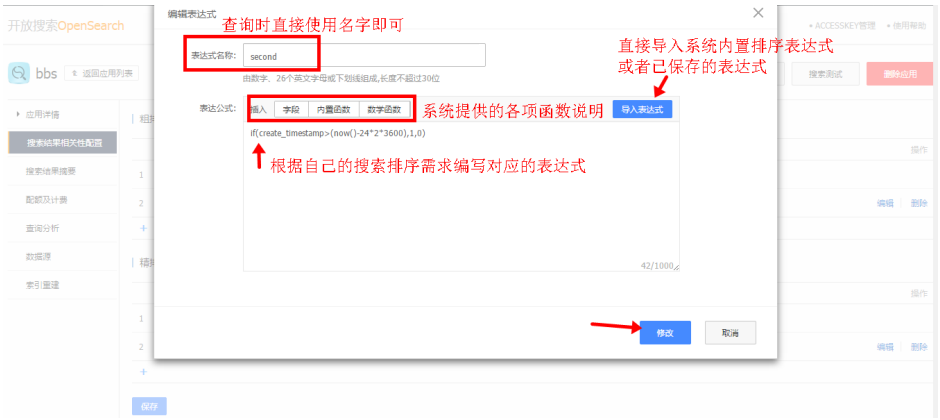
流程演示

主界面：

添加新粗排表达式或编辑现有表达式

添加新精排表达式或编辑现有表达式

编辑表达式内容



完成

搜索相关性函数

兼容特征及函数项

兼有function及feature的功能，可以同时在filter、sort及formula表达式中使用，**其中函数参数出现的文档字段必须配置为属性字段。**

注意：

粗精排表达式中建议一律使用**数值**或**数值类型字段**参与基本运算操作，例如算数，关系，逻辑，条件等运算操作，大部分函数都不支持字符串类型字段进行运算。

distance：获取两个点之间的球面距离，一般用于LBS的距离计算。

详细用法

distance(longitude_a, latitude_a, longitude_b, latitude_b, output_name)

参数
longitude_a：A的经度值，支持的参数类型为浮点型的字段名。
latitude_a：点A的纬度值，支持的参数类型为浮点型的字段名。
longitude_b：点B的经度值，支持的参数类型为浮点型的字段名，或者为用户查询串中kvpairs子句中设置的一个字段名（其值需要为浮点数）。
latitude_b：点B的纬度值，支持的参数类型为浮点型的字段名，或者为用户查询串中kvpairs子句中设置的一个字段名（其值需要为浮点数）。
outputname：如果需要在结果中返回距离值，可以通过制定outputname值得到，如果不需要，可以不指定。

> **注意**：
>
> outputname参数仅限于精排表达式中使用，filter及sort子句不支持。
>
> 设置outputname参数后，实际的距离值将在**variableValue**节点中展示，该节点只能在返回格式为fulljson中（config子句中format参数可以设置）才能得到。

返回值
float类型的实际距离值，单位为千米。

适用场景

场景1：

查找距离用户坐标（120.34256,30.56982）10公里内的外婆家，并按照距离由近及远排序（lon，lat为文档中记录商家的经纬度值，需要配置为属性字段）。

query=default:'外婆家'&&filter=distance(lon,lat,"120.34256","30.56982")<10&&sort=+distance(lon,lat,"120.34256","30.56982")

其中距离排序也可以采用如下方式实现，用户坐标通过kvpairs子句传递。

kvpairs=longitude_in_query:120.34256, latitude_in_query:30.56982

精排表达式为：

distance(longitude_in_doc, latitude_in_doc, longitude_in_query, latitude_in_query, distance_value)

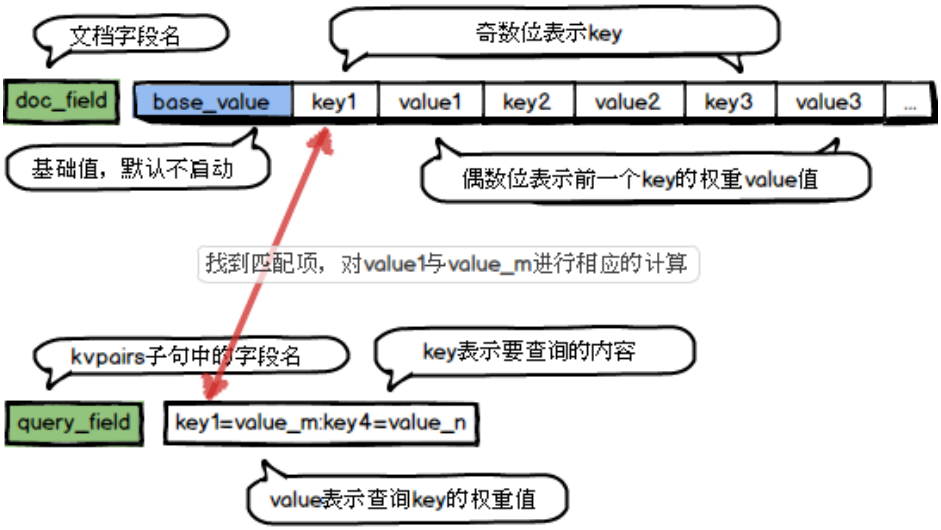
tag_match：用于对查询语句和文档做标签匹配，使用匹配结果对文档进行算分加权

场景概述

涉及query和文档匹配的很多需求都可以使用或者转化为tag_match来满足，对实现搜索个性化需求尤其有用。例如优先展现用户点赞过的店铺，或者优先展现用户喜欢的体育、娱乐类新闻等。

tag_match最基本的功能是在文档的某个ARRAY字段中存储一系列的key-value信息，然后在查询中通过kvpairs子句传递对应的key-value信息，tag_match就会去文档中寻找查询中指定的key，然后为每个匹配的key计算得到一个分数，再合并所有匹配key的分数得到一个最终分，该结果可以用来做算分加权或者过滤。

计算过程如下：



详细用法

常见用法：

tag_match(query_key, doc_field, kv_op, merge_op)

高级用法：

tag_match(query_key, doc_field, kv_op, merge_op, has_default, doc_kv, max_kv_count)

参数

query_key：

指定查询语句中用于匹配的字段key-value值，该字段需要通过kvpairs子句传递，key与value通过英文等号=分隔，多个key-value通过英文冒号：分隔。

例如：kvpairs=query_tags:10=0.67:960=0.85:1=48表示参数query_tags中包含3个元素：10、960、1，其对应的value分别是：0.67、0.85、48。

其中query也可以只为key的列表，例如：kvpairs=cats:10:960:1。

doc_field：

指定文档中存储key-value的字段名，该字段为整型或者浮点型的数组类型（如果是浮点型数组，则key的值匹配的时强转当int来处理，例如字段值为1.8，对应匹配的参数值应指定为1）。

数组的奇数位置是key，下一个相邻的偶数位置为key的value。即 [key0 value0 key1 value1 ...]

kv_op：

- 当query_key中的值与doc_field中的key匹配时，对二者的value所采取的操作，目前支持的操作符如下：
  - max（最大值）、min（最小值）、sum（求和）、avg（平均数）、mul（乘积）、query_value（query_key中该key对应的value值）、doc_value（文档中该key对应的value值）、number（常数）。

merge_op：

- 多个key匹配后会产生多个结果，merge_op指定了将这些结果进行如何操作，目前支持的操作类型如下：
  - max（最大值）、min（最小值）、sum（求和）、avg（平均值）。

has_default：

- 默认是false，表示不启动初始分值。
- 其值为true时说明doc_field的第一个值为默认值（类似base分值的概念），即 [init_score k0 v0 k1 v1...]。

doc_kv：

- 默认是true，表明doc_field字段中的值是以key-value对的形式存在。
- 其值为false则表示doc_field中只包含key信息，这种场景对doc需要存储标签，但是标签没有权重很方便。

max_kv_count：

- 因为查询中的key-value结构需要通过query传递，所有tag_match对query中能传递多少key-value有限制。
- 默认为50，可以通过这个参数将这个限制调大，但是最大不能超过5120。

返回值

double类型的具体分值，如果**has_default**参数值为**false**，并且没有配额的内容则为0。

如果需要返回int类型的结果，需要使用int_tag_match，该函数功能与参数与int_tag_match完全一致，但int_tag_match不能在排序表达式中使用。

适用场景

场景1：

一个大型的综合性论坛，帖子可以被打上各种各样的标签(搞笑，体育，新闻，音乐，科普..)。我们在推送给open_search的文档中，可以为每个标签赋予一个标签id（例如搞笑-1，体育-5， 新闻-3， 音乐-6..), 然后通过一个tag字段存储这些标签。 如果我们对帖子做过预处理，甚至能得到每个帖子每个标签的权重，例如一个搞笑体育新闻的帖子可以得到搞笑的权重为0.5，体育的权重为0.5，新闻权重为0.1，则这个帖子的tag字段的值为[1 0.5 5 0.5 3 0.1]对会员用户，通过长时间的积累，我们能获知每个用户的兴趣标签。

例如用户nba_fans对体育和搞笑很感兴趣，他对应的体育和搞笑标签的权重分别为0.6和0.3。那么这个用户查询时，我们就可以通过kv_pairs子句把这个信息加到query里面。假如这个kv_pairs子句名字为user_tag, 那么nba_fans的user_tag的值5=0.6:1=0.3。这样，我们只要在精排表达式中配置了tag_match(user_tag, tag, mul, sum), 我们就能够实现对用户感兴趣的帖子加权，把用户更感兴趣的帖子排到前面。

例如nba_fans搜索到上面那个帖子时，搞笑和体育这两个标签能够匹配到。通过指定kv_op参数为mul，我们会把query和doc中的值相乘，他们各自的计算分数分别为（体育：0.5 * 0.6 = 0.3, 搞笑：0.5 * 0.3 = 0.15)。通过指定merge_op参数为sum，我们会把体育和搞笑的分数加和（0.3+0.15 = 0.45），这个加和的分数会加到最终的排序分数上。这样，我们就能够实现了对这个用户感兴趣帖子的排序加权。

场景2：

商品可以具有多个属性标签，如1表示年轻人（年龄）、2表示中年人（年龄）、3表示小清新（风格）、4表示时尚（风格）、5表示女性（性别）、6表示男性（性别）等。

假设我们只想表示商品有没有某个标签，不想区分哪个标签更重要。这个标签通过options字段来保存。那么年轻时尚女性的衣服的options字段可以表示为[1 4 5], 注意这里

只有标签key，没有value。用户也都有自己的属性标签，和商品标签对应。例如年轻女性用户，历史成交中多购买小清新风格衣服。这该用户的查询可以写为user_options=1:3:5。注意这里kv_pair中也是只有标签key，没有value的。

要实现对符合用户标签喜好的商品加权，我们可以在formula中使用tag_match(user_options, options, 10, sum, false, false)。这里我们通过user_options和options指定了query和doc的标签信息。kv_op设为常数10，表示只要有标签匹配到，那么匹配的计算结果就是10。has_default为false，表示我们不需要初始值。doc_kv为false，表示我们doc中只存储了key信息，没有value。

这样，上面的年轻女用户查询到上面的衣服时，女性和年轻两个标签能够匹配上，这两个标签的计算结果都是10。通过sum这个merge_op，能够得到这件商品的最终加权分数为20。通过这种方式，即使我们没有标签的权重信息，也能够实现对匹配到的文档做排序加权。

注意事项

如果是用在 filter 或者 sort 子句中，则 query_key、kv_op、merge_op、has_default、doc_kv 必须使用**双引号**括起来，例如：

sort=-tag_match("user_options", options, "mul", "sum", "false", "true", 100)

再为 tag_match 配置 kvpairs子句传入对应参数值（否则会报语法错误）：

kvpairs=user_options:10

- tag_match 的 key 匹配都是通过整数比较来完成的，因此 query 和 doc 中的 key 都应该转换为整数形式，如果是浮点类型，tag_match 在比较时会强制转换为整数类型。

Function函数项

可以用在filter子句作为过滤和筛选条件，而返回值为数值型的fuction在sort子句中，用来做排序，**其中函数参数出现的文档字段必须配置为属性字段。**

注意：

粗精排表达式中建议一律使用**数值**或**数值类型字段**参与基本运算操作，例如算数，关系，逻辑，条件等运算操作，大部分函数都不支持字符串类型字段进行运算。

in/notin : 判断字段值是否（不）在指定列表中

详细用法
in(field, "number1|number2")

notin(field, "number1|number2")

参数
field：要判断的字段名，只支持INT及FLOAT类型，（不支持ARRAY及LITERAL、TEXT、模糊分词系列类型）
number列表：集合元素，多个值用’|’分隔，参数以字符串形式传入

返回值
true/false

适用场景
场景1：查询文档中包含“iphone”且type（int类型）为1或2或3的文档；
query=default:’iphone’ &&filter=in(type,"1|2|3")

场景2：查询文档中包含“iphone”且type（int类型）不为1或2或3的文档；
query=default:’iphone’ &&filter=notin(type,"1|2|3")

注意事项

- in(field, “number1|number2”)函数也等价于(field = number1) OR (field = number2)，但是前者的性能会更好，同理notin也类似。

fieldlen：获取literal类型字段长度

详细用法
fieldlen(field_name)

参数
field_name：要判断的字段名，可以为literal或者array类型。

返回值
字段内容长度，类型为int。如果字段为array类型，则返回数组元素个数。

适用场景
场景1：返回用户名usr_name不为空的文档
query=default:’关键词’ &&filter=fieldlen(usr_name)>0

in_polygon：判断某个点是否在某个多边形范围内，一般用于配送范围判断

详细用法
in_polygon(polygon_field, user_x_coordinate, user_y_coordinate,
has_multi_polygons="false")

参数
polygon_field: 表示商家配送范围的字段名，类型必须为DOUBLE_ARRAY, 字段值依次为配送多边形有序定点的x,y坐标（先x后y），顶点务必保证有序（顺时针、逆时针均可）！!如果有多个（N个）配送多边形，则第一个值表示多边形个数，第2~N+1的值表示后续每个多边形的顶点数（不是坐标数哦！！），第N+2值开始依次表示各多边形的顶点x,y坐标（**N的值域为[1,50]**）
user_x_coordinate: 用户的x坐标, double类型
user_y_coordinate: 用户的y坐标, double类型
has_multi_polygons：表示polygon_filed是否包含多个独立的多边形需要判断。默认为false，表示只有单一的多边形。

返回值
int，在多边形内返回第几个多边形匹配, 否则返回0。

适用场景
场景1：判断用户是否在商家的配送范围。如商家配送范围的字段为coordinates, 用户位置坐标为(120.307234, 39.294245)，则过滤在配送范围内的商家查询可写为：
query=default:’美食’ &&filter=in_polygon(coordinates, 120.307234, 39.294245)>0

注意事项

- 最多支持50个多边形，超过则跳过该文档的计算；
- 不支持有孔多边形，如环；
- 不支持多个分离部分的多边形；
- 坐标个数为0，表示没有坐标，返回0；
- 坐标个数为奇数个，则认为数据有误，返回0；
- 用户点位于多边形边上，则认为匹配成功，返回为1（或具体多边形下标）。
- 多边形插件计算量较大，对查询性能有影响，建议尽量控制顶点个数，具体值请根据自己实际情况进行测试得出。

in_query_polygon ： 判断文档中指定的点是否在用户指定的多边形范围内

详细用法
in_query_polygon(polygon_key, doc_point)

参数
polygon_key：kvpairs子句中定义的用户参数key，多边形顶点存储在对应的value中。类型必须为

DOUBLE_ARRAY,字段值依次为配送多边形有序定点的x,y坐标（先x后y），顶点务必保证有序（顺时针、逆时针均可）！！坐标之间用逗号分隔，格式为：xA,yA,xB,Yb。支持多个多边形，多边形与多边形之间通过分号（;）分隔。
doc_point：类型必须为DOUBLE_ARRAY，表示需要判断的点。只包含两个值，依次为点的x，y坐标

返回值
int，返回匹配到的第一个多边形的下标，没有匹配则返回0

适用场景
场景1：搜索银泰商圈（xA,yA,xB,Yb,xC,Yc;xD,yD,xE,yE,xF,yF,xG,yG）的外婆家，商家位置存放在point字段中
query=default:’外婆家’ &&filter=in_query_polygon("polygons", point)>0&&kvpairs=polygons:xA\,yA\,xB\,Yb\,xC\,Yc;xD\,yD\,xE\,yE\,xF\,yF\,xG\,yG

multi_attr : 返回数组字段指定位置的值

详细用法
multi_attr(field, pos, default_value=0|"")

参数
field: 查询的字段名，必须为ARRAY数组类型，且必须配置为属性字段
pos: 整形常数或整形字段，需要配置为属性字段，，下标从0开始
default_value: 可选，字符串常量。表示如果指定pos的值不存在时，返回default_value

返回值
类型和field保持一致

适用场景
场景1：商品有多个价格[市场价、折扣价、销售价]，存到prices字段中。查询销售价小于1000的手机 query=deault:’手机’ &&filter=multi_attr(price,2)<1000

bit_struct: 将INT_ARRAY字段值进行自定义分组并允许对分组值进行指定operation计算

详细用法

bit_struct(doc_field, “$struct_definition”，operation, ...)

参数

doc_field：

- 是一个INT_ARRAY类型的字段名。

$struct_definition：

用于把int的值拆分成多个维度的信息。每一维的分组用int中的起始bit位置和结束bit位置指定，使用横线“-”分隔，bit位置从值的高端开始算起，从0开始，最大不能超过63。多个分组用逗号”，”分隔。每个分组有一个编号，编号从1开始算起。

举例：假设用户需要把int拆分成3个维度的信息， bit0到bit9代表一个值（用$1表示），bit10到bit48代表一个值（用$2表示），bit49到bit60代表一个值（用$3表示），则该参数可写成:”0-9,10-48,49-60”。

operation：

定义计算过程,最少定义1个，最多定义5个,每个operation会有一个编号，这个编号是接着struct_definition中的编号开始递增，当需要定义多个operation时，后面的operation要用到前面计算过的operation的返回值，这时候就可以用到给operation分配的编号了。

operation 可以定义的操作有：

- “equal,$m,$n”：判断$m代表的值和$n代表的值是否相等，相等返回true,否则返回false。
- “overlap,$m,$n,$k,$p”：判断($m,$n)和($k,$p）定义的范围在数轴上是否相交。相交时返回true，否则返回false

“and,$m,$n,…”：返回$m, $n,..等做and(&&）的结果。

注意：
上面的这3个操作的参数也可以是整数数字。 例如 “equal,$1,1”

返回值

int，返回最后一个operation第一次为true时对应的doc_field中的数组下标(从0开始)。若doc_field中没有满足operation指定的要求的值，则返回-1。

场景举例

查询给定时间段在营业的店铺有哪些
假定用户文档中有一个int_array类型的字段open_time，每个值表示一段营业时间，将int的高32位表示起始时间，低32位表示结束时间，如果要查询下午14点到15:30点营业的

店铺，可以将时间转换为从当天0点开始，按分钟为单位的时间段, 则下午14点到15:30表示为（840,930）,则查询中filter子句可以写为：
filter=bit_struct(open_time, "0-31,32-63","overlap,$1,$2,840,930")!=-1

查询未来某一天,某个餐点(早，中，晚)，可以提供Pmin到Pmax人数就餐的店铺
假设用户文档中有一个int_array类型的字段book_info，对于该字段中的一个值，0-7位表示日期，8-15位餐点，16-41位表示最小人数，42-63位表示最大人数。查询明天（用1表示）晚上（用3表示）能服务3-5个人的店铺，则filter子句可以写为：
filter=bit_struct(book_info,”0-7,8-15,16-41,42-63”,
“equal,$1,1”,”equal,$2,3”,”overlap,$3,$4,3,5”,”and,$5,$6,$7”)!=-1
这里$1表示book_info中0-7位代表的值，
$2表示book_info中8-15位代表的值
$3表示book_info中16-41位代表的值
$4表示book_info中42-63位代表的值
$5代表operation “equal,$1,1”的返回值
$6代表 operation”equal,$2,3”的返回值
$7代表operation “overlap,$3,$4,3,5”的返回值
返回$5,$6,$7代表的值做and(逻辑与)后第一次为true时候的值 在book_info中对应的数组下标

查询下午14点到15:30表示为（840,930）之间，库存>10的店铺有哪些？
因为bit_struct返回的是下标，所以他可以和multi_attr函数一起配合使用，取另外一个array类型字段对应下标的值。如该例，可以在查询语句中使用：
filter=multi_attr(store, bit_struct(dispatch_time,"0-31,32-63", "equal,$1,840", "equal,$2,930", "and,$3,$4"))>10
dispatch_time是文档中有一个多值INT的字段，用于存储商户的配送时间。将时间转换为从当天0点开始，按分钟为单位的时间段, 则下午14点到15:30表示为（840,930）
store是一个int_array字段，与dispatch_time的时间段分别对应，表示该时间段的库存量。

注意事项

- 更多介绍请参考这里。

Feature功能

可以用到排序表达式中（大部分仅支持精排表达式），可以通过各种语法及语句的组合得到强大的排序功能，**其中函数参数出现的文档字段必须创建为索引。**

注意：

粗精排表达式中建议一律使用**数值**或**数值类型字段**参与基本运算操作，例如算数，关系，逻辑，条件等运

算操作，大部分函数都不支持字符串类型字段进行运算。

static_bm25：静态文本相关性，用于衡量query与文档的匹配度

详细用法
static_bm25()

参数
无

返回值
float，值域为[0,1]

适用场景
场景1：在粗排中指定文本分；
在粗排表达式中指定static_bm25()

注意事项

- 可以用在粗排表达式

timeliness：时效分，用于衡量文档的新旧程度

详细用法
timeliness(pubtime)

参数
pubtime：要评估的字段，类型必须为int，单位为秒。

返回值
float，值域为[0,1]，值越大表示时效性越好。若大于当前时间则返回0。

适用场景
场景1：在精排中指定create_timestamp字段的时效性；
在粗排表达式中指定timeliness(create_timestamp)

注意事项

- pubtime字段必须配置为属性字段；
- 可以用在粗排和精排表达式。

timeliness_ms：时效分，用于衡量文档的新旧程度

详细用法
timeliness_ms(pubtime)

参数
pubtime：要评估的字段，类型必须为int，单位为毫秒。

返回值
float，值域为[0,1]，值越大表示时效性越好。若大于当前时间则返回0。

适用场景
场景1：在精排中指定create_timestamp字段的时效性；
在粗排表达式中指定timeliness_ms(create_timestamp)

注意事项

- pubtime字段必须配置为属性字段；
- 可以用在粗排和精排表达式

normalize ：归一化函数，根据不同的算分将数值归一化至[0, 1]

场景概述
相关性计算过程中，一篇doc的好坏需要从不同的维度衡量。而各个维度的分数值域可能不同，比如网页点击数可能是成百上千万，网页的文本相关性分数在[0, 1]之间，它们之间没有可比性。为了在公式中使用这些元素，需要将不同的分数归一化至同一个值域区间，而normalize为这种归一化提供了一种简便的方法。normlize支持三种归一化方法：线性函数转化、对数函数转化、反正切函数转化。根据传入参数的不同，normalize自动选择不同的归一化方法。如果只指定value参数，normalize使用反正切函数转化，如果指定了value和max参数，normalize使用对数函数转化，如果指定了value、max和min，normalize使用线性函数转化。

详细用法：
normalize(value, max, min)

参数
value：需要做归一化的值，支持double类型的浮点数，该值可以来自文档中的字段或者其他表达式
max：value的最大值，可选，支持double类型的浮点数
min：value的最小值，可选，支持double类型的浮点数

返回值
double，[0, 1]之间的值。

适用场景
场景1：对price字段做归一化，但是不知道price的值域，可以使用如下公式进行归一化
normalize(price)

场景2：对price字段做归一化，但是只知道price的最大值为100，可以使用如下公式进行归一化
normalize(price, 100)

场景3：对price字段做归一化，并且知道price的最大值为100，最小值为1，可以使用如下公式进行归一化
normalize(price, 100, 1)

场景4：将distance函数的结果归一化至[0, 1]
normalize(distance(longitude_in_doc, latitude_in_doc, longtitude_in_query, latitude_in_query))

注意事项

- 使用反正切函数进行归一化时，如果value小于0，归一化后的值为0
- 使用对数函数进行归一化时，max的值要大于1
- 使用线性函数进行归一化时，max要大于min

gauss_decay：使用高斯函数，根据数值和给定的起始点之间的距离，计算其衰减程度

详细用法
gauss_decay(origin, value, scale, decay, offset)

参数
origin：衰减函数的起始点，支持double类型的浮点数
value：需要计算衰减程度的值，支持double类型的浮点数，该值可以来自用户字段或者其他表达式
scale：衰减程度，支持double类型的浮点数
decay：当距离为scale时的衰减程度，支持double类型的浮点数，可选，默认值为0.000001
offset：当距离大于offset时才开始计算衰减程度，支持double类型的浮点数，可选，默认值为0

返回值
返回值为double，区间为[0, 1]

适用场景
场景1：查找距离用户最近的酒店，按照距离由近到远排序，并且认为距离小于100m的酒店不用做

区分, longitude_in_doc和latitude_in_doc为酒店的经纬度，longtitude_in_query和latitude_in_query为用户的经纬度
gauss_decay(0, distance(longitude_in_doc, latitude_in_doc, longtitude_in_query, latitude_in_query), 5, 0.000001, 0.1)

场景2：查找2000元左右的手机，并且如果价格小于1500或者大于2500时，文档算分为0，文档中手机价格为price，kvpairs=price_key:2000，公式如下：
gauss_decay(kvpairs_value(price_key, FLOAT), price, 500)

注意事项

- 如果scale小于或者等于0，衰减函数默认返回0
- 如果decay大于或者等于1，衰减函数默认返回1
- 如果decay小于或者等于0，默认将decay设置为0.000001
- 如果offset小于0，默认将offset设置为0

exp_decay：使用指数函数，根据数值和给定的起始点之间的距离，计算其衰减程度

详细用法
exp_decay(origin, value, scale, decay, offset)

参数
origin：衰减函数的起始点，支持double类型的浮点数
value：需要计算衰减程度的值，支持double类型的浮点数，该值可以来自用户字段或者其他表达式
scale：衰减程度，支持double类型的浮点数
decay：当距离为scale时的衰减程度，支持double类型的浮点数，可选，默认值为0.000001
offset：当距离大于offset时才开始计算衰减程度，支持double类型的浮点数，可选，默认值为0

返回值
返回值为double，区间为[0, 1]

适用场景
同gauss_decay，只是衰减算法不同

注意事项

- 如果scale小于或者等于0，衰减函数默认返回0
- 如果decay大于或者等于1，衰减函数默认返回1
- 如果decay小于或者等于0，默认将decay设置为0.000001
- 如果offset小于0，默认将offset设置为0

linear_decay：使用线性函数，根据数值和给定的起始点之间的距离，计算其衰减程度

详细用法
linear_decay(origin, value, scale, decay, offset)

参数
origin：衰减函数的起始点，支持double类型的浮点数
value：需要计算衰减程度的值，支持double类型的浮点数，该值可以来自用户字段或者其他表达式
scale：衰减程度，支持double类型的浮点数
decay：当距离为scale时的衰减程度，支持double类型的浮点数，可选，默认值为0.000001
offset：当距离大于offset时才开始计算衰减程度，支持double类型的浮点数，可选，默认值为0

返回值
返回值为double，区间为[0, 1]

适用场景
同gauss_decay，只是衰减算法不同

注意事项

- 如果scale小于或者等于0，衰减函数默认返回0
- 如果decay大于或者等于1，衰减函数默认返回1
- 如果decay小于或者等于0，默认将decay设置为0.000001
- 如果offset小于0，默认将offset设置为0

exact_match_boost ：获取查询中用户指定的查询词权重最大值

详细用法
exact_match_boost()

参数
无

返回值
int，值域为[0, 99]

适用场景
场景1：查询为query=default:'开放搜索'^60 OR default:'opensearch'^50，希望按照实际匹配词boost权重来排序。如如果文档A包含“开放搜索”，文档B包含“opensearch”，则文档A排到文

档B前面。
粗排表达式为：exact_match_boost() 精排表达式为空。 //精排为空，默认按照粗排表达式分值来排序。

注意事项

- 对于没有指定boost的查询词，默认boost权重值为99。
- 支持不同索引指定权重。
- 配置精排表达式后会有影响。

first_phase_score ： 获取粗排表达式最终计算分值

详细用法
first_phase_score()

参数
无

返回值
float

适用场景
场景1：粗排表达式为exact_match_boost()，精排为exact_match_boost()与text_relevance(title)，且二者权重为3:1。
粗排表达式：exact_match_boost()
精排表达式：first_phase_score()*0.01*3+text_relevance(title) //直接使用first_phase_score()而exact_match_boostce()可以减少计算量，提高检索性能。

注意事项

- 多个OR查询情况下，OR个数及查询召回文档数都对性能影响很大，需要根据实际场景进行详细的测试和优化。

text_relevance：关键词在字段上的文本匹配度

详细用法
text_relevance(field_name)

参数
field_name：字段名，该字段需要为中文基础分词、中文基础分词、自定义分词、单字分词等类型，并且配置了索引字段。

返回值
float，值域为[0,1]

适用场景
场景1：在精排中对title和body进行文本算分，权重比为3:1
text_relevance(title)*3+text_relevance(body)

注意事项

- 主要衡量角度：命中词在query中所占比重；命中词在字段中所占比重；命中词在字段中出现的频率；字段中命中词之间的顺序关系与query中命中词之间的顺序关系。
- 该feature目前只用于精排排序。

fieldterm_proximity：用来表示关键词分词词组在字段上的紧密程度

详细用法
fieldterm_proximity(field_name)

参数
field_name：该字段需要为TEXT、中文基础分词、自定义分词、单字分词等类型，并且建立了可搜索

返回值
float，值域为[0,1]

适用场景
场景1：在精排阶段计算query在title和body的紧密度，并且title字段的紧密度在排序中起主导作用，则在创建精排公式时公式内容可以写为：
fieldterm_proximity(title)*10 + fieldterm_proximity(body)

注意事项

- 主要衡量角度：命中词在字段中的距离，命中词在字段中的相互顺序。
- 该feature目前只用于精排排序，且包含在text_relevance()中，即普通场景下二者无需共用。

kvpairs_value：获取查询串中kvpairs子句中指定字段的值

详细用法
kvpairs_value(query_key, type)

参数
query_key：要返回的kvpairs子句中的字段名

type：kvpairs中query_key字段值的类型，目前支持的类型如下：INT，FLOAT，DOUBlE。

用法示例
场景1：查询串中kvpairs子句中设置了query_key:10，10是整数类型,期望公式中取出query_key的value，公式可以写为:
kvpairs_value(query_key, INT)

场景2：查询串中kvpairs子句中设置了query_key:10.1, 10.1是float类型,期望公式中取出query_key的value，公式可以写为:
kvpairs_value(query_key, FLOAT)

场景3：查询串中kvpairs子句中设置了query_key:10.12, 10.12是double类型，期望公式中取出query_key的value，公式可以写为:
kvpairs_value(query_key, DOUBLE)

query_term_count : 返回查询词分词后词组个数

详细用法：
query_term_count()

参数：
无

返回值：
int

适用场景：
场景1：根据查询词中term的个数做不同的处理；
if (query_term_count() > 10, 0.5, 1)

注意事项：

- 仅用于精排表达式

query_term_match_count ：获取查询词中（在某个字段上）命中文档的词组个数

详细用法：
query_term_match_count(field_name)

参数：
field_name： 非必选参数，要统计的字段名，该字段类型可以是TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段。若不指定该参数，则默认返回全部字段命中的词组个数。

返回值：
int

适用场景：
场景1：根据查询词在文档中title字段上命中的词组个数做不同的处理；
if (query_term_match_count(title) > 10, 0.5, 1)

场景2：根据查询词中命中的词组个数做不同的处理；
if (query_term_match_count() > 10, 0.5, 1)

注意事项：

- 可以用于精排表达式
- 统计的时查询词中命中的分词词组个数，重复的词组会计算多次

field_term_match_count ：获取文档中某个字段与查询词匹配的词组个数

详细用法：
field_term_match_count(field_name)

参数：
field_name： 要统计的字段名，该字段类型可以是TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段。

返回值：
int

适用场景：
场景1：根据字段中匹配的分词词组的个数做不同的处理
if (field_term_match_count(title) > 5, 0.8, 0.6)

注意事项：

- 可以用于精排表达式
- 统计的是字段中命中的分词词组的个数，重复的词组会计算多次

query_match_ratio ：获取查询词中（在某个字段上）命中词组个数与总词组个数的比值

详细用法：
query_match_ratio(field_name)

参数：
field_name， 非必选参数，要统计字段名，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段

返回值：
float，值域为[0, 1]

适用场景：
场景1：判断查询词中的词组是否全部命中文档
if (query_match_ratio() > 0.999, 1, 0)

场景2：判断查询词中的词组是否全部命中文档的title字段
if (query_match_ratio(title) > 0.999, 1, 0)

注意事项：

- 可以用于精排表达式

field_match_ratio： 获取某字段上与查询词匹配的分词词组个数与该字段总词组个数的比值

详细用法：
field_match_ratio(field_name)

参数：
field_name：要统计的字段名，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段

返回值：

float，值域为[0, 1]

适用场景：
场景1：在精排阶段计算title和body与查询词的匹配程度
field_match_ratio(title)*10 + field_match_ratio(body)

注意事项：

- 可以用于精排表达式
- 该feature可以从一定程度上反应出field与query的匹配程度。

field_length：获取某个字段上的分词词组个数

详细用法：
field_length(field_name)

参数：
field_name：要获取的字段名，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分词、模糊分词类型，并且配置了索引字段。

返回值：
int

适用场景：
场景1：根据字段分词词组个数设置不同的权重
if (field_length(title) > 200, 0.3, 0.7)

注意事项：

- 可以用于精排表达式

query_min_slide_window：查询词在某个字段上命中的分词词组个数与该词组在字段上最小窗口的比值

详细用法：
query_min_slide_window(field_name, in_order=false)

参数：
field_name：要统计的字段，该字段需要为TEXT、中文基础分词、自定义分词、单字分词、英文分

词、模糊分词类型，并且配置了索引字段。
in_order：true|false，默认为false。表示进行滑动窗口比较时，窗口中词组的顺序是否必须和查询词中的保持一致。

返回值：
float，值域为[0, 1]

适用场景：
场景1：计算查询词在title上的最小窗口
query_min_slide_window(title)

场景2：判断title字段中是否存在于查询词中相同的子序列
if(query_min_slide_window(title, true) > 0.99, 1, 0)

注意事项：

- 可以用于精排表达式；
- 从字面上衡量query在field_name字段上紧密度情况；
- 影响滑动窗口计算的有两个因素，query在field_name字段上命中的term的个数和包含这些term的最小窗口。

搜索结果摘要

一般文档内容会比较长，而在实际展示搜索结果的时候，不可能完全展示出来。这时候就需要做摘要及飘红设置。系统会截取包含搜索结果的几个片段，供用户了解具体匹配内容，以快速判断是否是自己想要的结果。

允许用户对搜索结果的展示效果进行自定义设置，设置完成后，用户在调用 API 时，系统会自动获取用户配置，并添加到查询 query 中，无需用户再次传入。当然也可以在 API 参数中通过 summary 参数进行具体查询的控制。

注意

不支持摘要与飘红分开配置。

若配置多个摘要字段，并且对应摘要字段值中有包含索引中指定搜索关键词的分词，则这些摘要字段中对应分词内容都会摘要飘红。

若对应用中某个字段分别创建不同分词类型，例如同时创建了中文基础及单字分词，此时中文单字分

词摘要飘红会有问题，该摘要飘红内容只会匹配中文基础分词，或出现内容飘红不对。

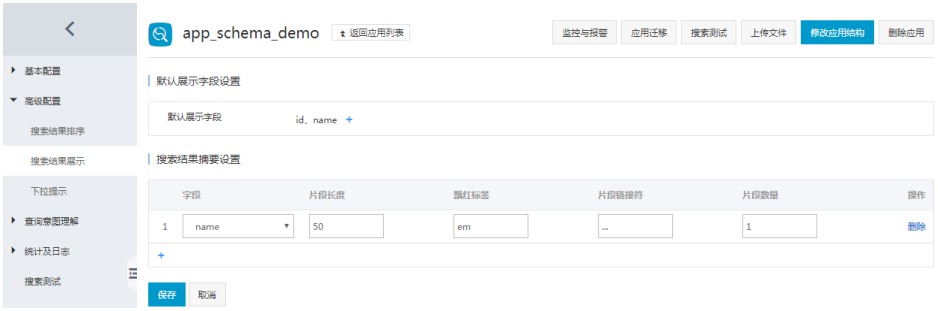
同一个请求query中，设置2种及以上不同类型分词索引进行搜索召回，会导致不飘红或飘红异常。

主要参数

- **字段**：需要配置摘要的字段
- **片段长度**：表示摘要长度
- **飘红标签**：关键词飘红的html标签
- **片段连接符**：每个片段之间的连接符
- **片段数量**：在摘要长度内需要几个片段

摘要配置

以下示例主要针对名称字段，做摘要飘红配置。



摘要飘红效果展示：

查询分析

为解决目前用户长尾query召回少、搜索词填写错误无法召回、输入拼音无法召回等问题，增加查询分析功能，提升用户的搜索体验。

允许用户针对不同的索引字段制定不同的查询分析规则，一条查询分析规则包含一个或多个功能项。

可配置的分词类型

目前查询分析中可配置的分词，只支持TEXT分词类别中的，中文基础分词 和 英文(去词根分词) 这2种，其它字段分词类型暂不支持。

流程演示

添加一个未上线规则：

选择该规则的功能：

修改适用范围（目前仅TEXT类型的索引字段可以配置该功能）。规则创建完毕后，可以通过查询中显式的指定qp=test1的方式进行搜索效果调试。调试无误后，可以“添加至上线”。

添加到已上线的规则会默认对所有查询语句起作用（系统会默认给查询拼上qp=test1，除非用户自己显式的拼接了qp参数）

下拉提示

功能介绍

下拉提示是搜索服务的基础功能，在用户输入查询词的过程中，智能推荐候选query，减少用户输入，帮助用户尽快找到想要的内容。

下拉提示在实现了中文前缀，拼音全拼，拼音首字母简拼查询等通用功能的基础上，实现了基于用户文档内容的query智能识别。用户通过控制台的简单配置，就能拥有专属的定制下拉提示。

例如对连衣裙这个query，用户既可以通过连衣查询到，也可以通过拼音lianyi，首字母简拼lyq查询。此外，下拉提示还提供了黑名单，推荐词条功能，让用户进一步控制下拉提示的结果，实现更灵活的定制。

支持应用

- 此篇文档的下拉提示功能，是针对老高级版应用类型的，新高级版查看下一篇新下拉提示文档。
- 标准版应用类型，不支持下拉提示。

黑名单

如果用户在下拉提示结果中发现了不想要的结果（比如一些黄色暴力的结果），可以通过把这些结果添加到黑名单中，实现对下拉提示结果的干预。

- 黑名单中的词条会从下拉提示索引中删除，任何词都不会再使下拉提示推荐该结果。
- 编辑黑名单之后，要点击控制台的“生效下拉提示”，等待状态变为“已生效”之后，新添加的黑名单才会起作用。
- 现在黑名单是完整匹配，只有黑名单中相同关键词才会删除。后续会推出pattern匹配，只要满足黑名单指定模式的结果都会删除。
- 黑名单最多添加500条。

推荐词条

系统会把推荐名单这些query优先放到下拉结果的最前面。用户可以向推荐词条中添加opensearch没有识别的query，排序靠后的优质query，一些导流query等。

- 推荐词条和黑名单一样，新添加的词条需要在控制台点击“生效下拉提示”并且状态变为“已生效”之后才会起作用。
- 黑名单和推荐词条冲突时，黑名单优先级更高，即该词条不会出现在下拉提示结果中
- 推荐词条最多添加500条。

注意事项

- 目前下拉提示为免费功能，默认最大 QPS 为当前应用容量中 QPS 峰值的 3倍流量配额。超过会被拒绝，请修改搜索QPS峰值。
- 目前下拉提示仅支持 TEXT 及 SHORT_TEXT 类型，自定义分词因为分词不一致，效果跟实际可能有差别。
- 下拉提示中需添加的字段必须要创建为索引，否则无法选择对应字段。
- **下拉提示会根据特征及算分，只抽取配置字段中有意义的词汇。**
- **下拉提示只会抽取应用中部分文档（最多百万级别）参与下拉提示候选集。**
- **下拉提示，不支持增量索引构建，目前是通过全量构建生效，因此增量推送到应用中的文档，下拉提示将无法召回。**
- **下拉提示主要是提高用户输入效率，搜索时最多支持18个字节的搜索召回，超过则无法召回。**
- 字段内容在60个字节内，会进行原值展示，超过则进行语义单元抽取（最多512个字节，超过则截断）。
- **下拉提示目前会对标点符号、不可见字符做过滤。**
- 下拉提示字段选取建议：
    - 尽量和使用下拉提示结果的索引的字段保持一致。
    - 尽量使用内容简洁，不要包含太多的html标签，富文本内容等和文档主题不相关的信息。
    - 尽量使用内容有差异化的字段集合。

流程演示

点击管理进入应用详情页，在“高级配置”中找到下拉提示，点击“+”进行规则添加：

填写规则内容：

规则创建完毕后，需要点击“生效下拉提示”：

生效完毕后可以进行效果测试：

最后调用下拉提示API接口嵌入到自己的搜索页面即可。

新下拉提示

功能介绍

下拉提示是搜索服务的基础功能，在用户输入查询词的过程中，智能推荐候选query，提高用户输入效率，帮助用户尽快找到想要的内容。

下拉提示实现了基于用户文档内容的query智能识别通过中，可以通过中文前缀，拼音全拼，拼音首字母简拼查询以及汉字加拼音等查询下拉提示的候选query。

例如，**连衣裙** 这个query，可以通过如下方式查询得到：

- 中文前缀：连，连衣;
- 全拼前缀：l, li, lian, lianyi, lianyiqun, ...
- 简拼前缀：l, ly, lyq;
- 汉字加拼音： 连yi, 连衣qun；

但不支持下拉方式查询：

- 拼音加汉字： lian衣， lian衣裙，连yi裙
- 原文：连衣裙

此外，用户可以通过干预词条对下拉提示的数据进行干预。

数据来源

下拉提示数据目前主要来源于应用的文档。每个下拉提示最多可以从一个应用中选择3个文本类型的字段作为数据来源。处理时，系统会选取应用中的部分文档（百万级别），对这些文档中被选中的字段按照指定的规则进行处理，生成下拉提示的候选数据。之后再按一定规则保留一定量的数据作为下拉提是候选query（百万级别）。

query生成规则

目前，系统支持两种规则生成候选query：抽取和原值保留。

抽取

对字段的内容进行一定的处理，抽取有意义的term进行组合，得到一些候选query。这种方式尽量保证生成的候选query能召回对应的文档。

原值保留

顾名思义，该规则对字段内容不做任何处理，直接将其作为下拉提示的候选query。但当字段内容超过一定限制（60字节）时，将字段内容进行截断，保留前60个字节的内容作为候选query。该规则适用于字段内容较短且含义明确。

干预词条

目前下拉提示支持对候选query进行干预，包括推荐名单和黑名单。

黑名单

在黑名单中的query，就不会再出现在下拉提示的结果中。因此，当下拉提示结果中出现一些业务上不想要的结果时，可通过将这些结果添加到黑名单中，就可以实现对这些结果的屏蔽。目前，黑名单只支持精确匹配。

后续将会增加pattern匹配的功能。

推荐名单

在推荐名单里的query，系统会将其排在下拉结果的前面。因此，当某些优质query没有被系统识别出来，或者排序靠后时，通过将其加到推荐名单中，达到更好的下拉提示效果。

配额说明

- 目前下拉提示为免费功能，因此计算资源和存储资源为系统统一设置。每个下拉提示的计算资源约为100QPS，存储资源约200W个候选query；

注意事项

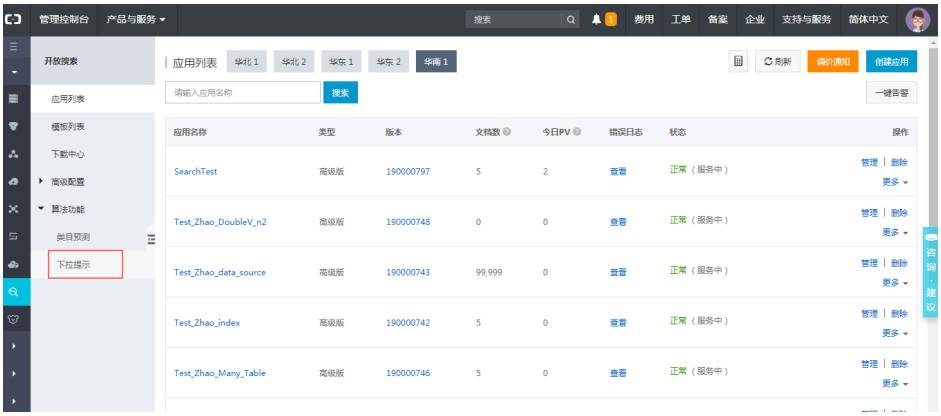
- 下拉提示仅支持从一个应用中选择创建了索引的TEXT、SHORT_TEXT类型的字段作为数据来源；
- 修改应用结构时，不能修改下拉提示数据来源的字段；
- 删除应用时，对应的下拉提示也会被删掉；
- 下拉提示搜索时，query参数长度最大支持18字节，否则会报错，返回无结果；
- 下拉提示搜索时，hit参数的取值范围为(0, 10]，否则按默认10处理，且会返回一个错误（例：当hit设置了0或-1或11时，默认按10处理并返回一个错误。）；
- 推荐名单中的query条数不能超过500；
- 黑名单中的query条数不能超过500；
- 推荐名单和黑名单数据有冲突时，黑名单优先级更高；
- 推荐名单和白名单的修改都需要重新训练模型后才能生效;
- 下拉提示的数据更新，主要通过用户手动触发，或者系统默认的周期性更新；数据更新记录可以通过页面上线的模型训练记录查看；
- 下拉提示的训练的时间，和应用的数据量、当时的系统负载有关；如果长时间（大于半个小时）没有训练结束，请联系我们；

使用建议

- 选择内容简洁，和文档主题相关的字段；
- 合理使用原值保留和抽取规则；
- 查询结果中，suggestions是本次查询的结果，errors表示本次查询是否有错误发生。errors不为空，不代表suggestions为空。因此，解析时结果时，都得通过suggestions是否为空来判断是否无数据展示。

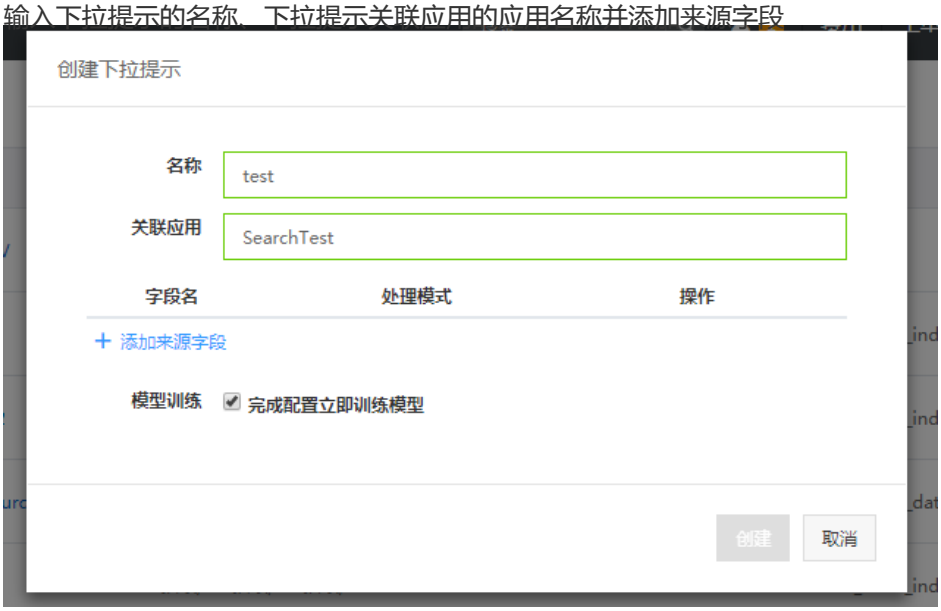
流程演示

点击应用列表——算法功能：下拉提示（下图的搜索测试均以SearchTest应用为例）

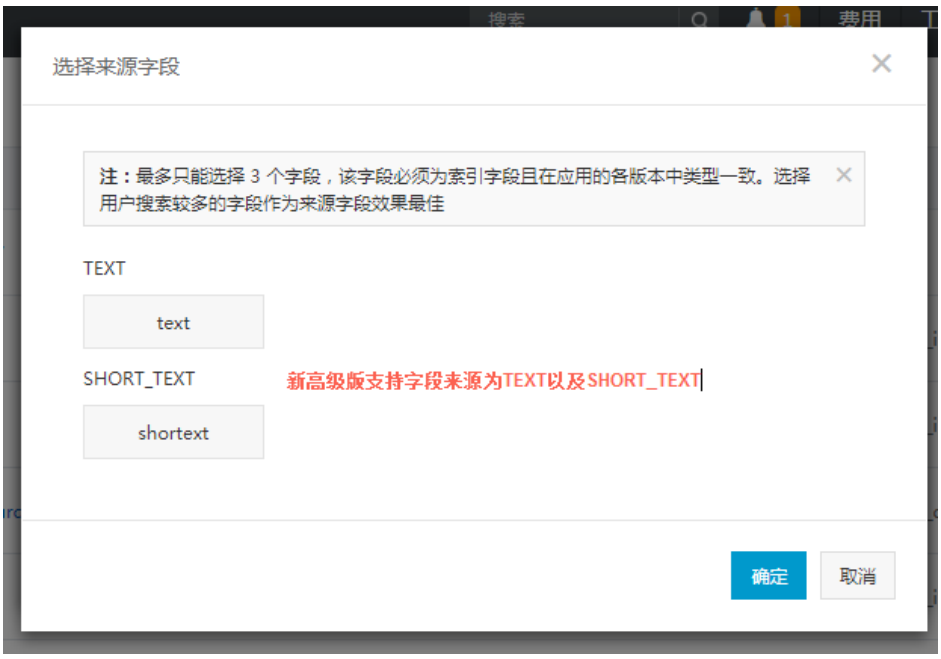


点击“创建下拉提示”

输入下拉提示的名称、下拉提示关联应用的应用名称并添加来源字段



选择来源字段（来源字段可以支持TEXT以及SHORT_TEXT两种类型）

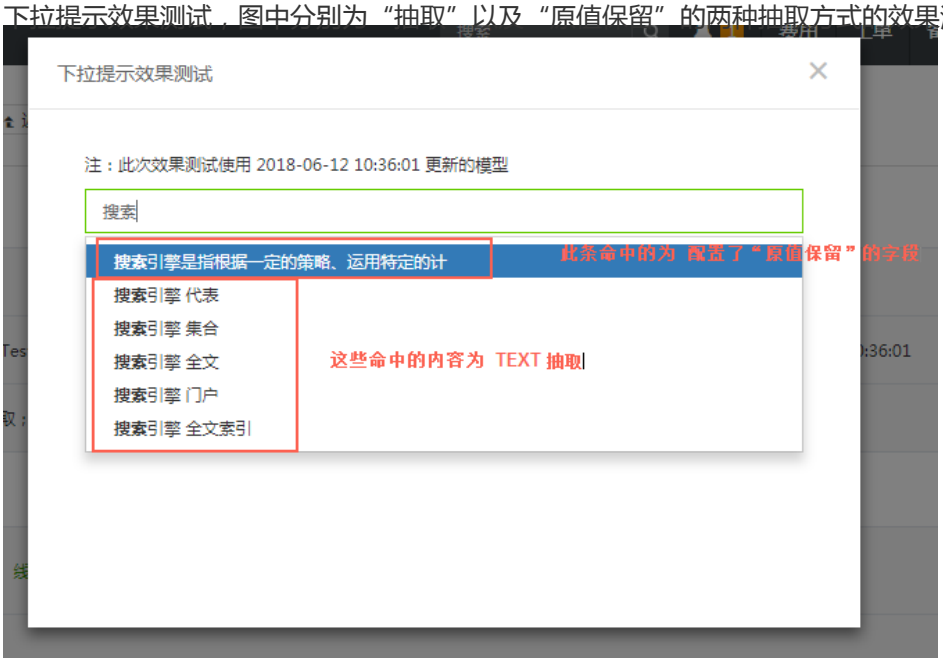


如图选择了两种处理方式，抽取以及原值保留作为样例进行下拉提示测试

点击“确定”后，勾选完成后立即训练（或者不勾选手动训练），保存下拉提示配置生效

下拉提示创建后，需要20-30分钟左右的训练时间

下拉提示效果测试，图中分别为“抽取”以及“原值保留”的两种抽取方式的效果测试



注意：

已经使用老版本下拉提示的应用（老高级版）不能使用新下拉提示，您可以迁移到新高级版应用再使用。

新下拉提示建议使用新的sdk/api进行调用

新下拉提示 Java SDK 和 Demo

SDK

aliyun-sdk-opensearch 3.2.0

```
<dependency>
<groupId>com.aliyun.opensearch</groupId>
<artifactId>aliyun-sdk-opensearch</artifactId>
<version>3.2.0</version>
</dependency>
```

Demo

API: GET v3/openapi/suggestions/{suggestion_name}/actions/search?hit=10&query={your_query}

Java SDK demo 如下

```
package com.example.opensearch;

import com.aliyun.opensearch.OpenSearchClient;
import com.aliyun.opensearch.SuggestionClient;
import com.aliyun.opensearch.sdk.generated.OpenSearch;

public class SuggestionSearch {
public static void main(String[] args) {
final String accesskey = "填入accesskey信息";
final String secret = "填入secret信息";
final String host = "填入下拉提示关联应用所在区域的host";

final String suggestionName = "填入下拉提示名称";
final String query = "填入查询词";
final byte hits = 8; // 最大返回条数

final OpenSearch opensearch = new OpenSearch(accesskey, secret, host);
final OpenSearchClient serviceClient = new OpenSearchClient(opensearch);
final SuggestionClient suggestionClient = new SuggestionClient(suggestionName, serviceClient);

suggestionClient.setQuery(query);
suggestionClient.setHits(hits);

String result;

try {
result = suggestionClient.search();
} catch (Exception e) {
e.printStackTrace();
return;
}

System.out.println(result);
```

```
}
}
```

自定义分词器

功能介绍

分词是搜索引擎中一个基础但重要的组件，分词的结果直接影响搜索效果。由于业务场景的多样，同一个短语在不同的业务、不同的语境下，其语义可能会不一样，期望分词的结果也不一样。为此，OpenSearch除了提供面向通用领域的基础分词器外，还提供了面向特定领域的分词器，如面向电商领域的电商分词器等。为了更好的满足用户的业务需求，OpenSearch可以让用户在系统提供的基础分词器的基础上，通过结合干预词条的形式创建自定义分词器。在应用的索引字段的分词器中选择使用相应的分词器，以达到干预索引和查询时分词结果，确保搜索结果的质量。

干预词条

目前，系统支持两种类型的干预词条：语义实体和语义切分。

语义实体干预词条

这类干预词条主要用于一些系统尚未识别的实体词，干预后，该词的切分总是能保持一致，不受其所在的上下文影响。

- 词条格式：

```
  实体词1
实体词2
```

- 示例：

```
  分词器
开放搜索
```

干预后，“分词器”和“开放搜索”不管出现什么样的上下文中，都会被当做独立的语义实体，系统对它们还会进一步切分，例如：“分词器”可能会保留原词，而“开放搜索”可能会被进一步切分成“开放“和“搜索”两个更细粒度的词。

语义切分干预词条

这种类型的干预词条用于指定在特定上下文中的短语的切分方式，而不影响该短语在其他上下文中的切分方式。

- 词条格式：

```
短语 => 短 语
```

- 示例：

```
语义切分干预词条 => 语义 切分 干预词条
```

干预后，当语句中完整的出现短语“语义切分干预词条”时，系统首先会按照词条对其切分成“语义”、“切分”和“干预词条”三个子短语，然后对子短语分别进行更细粒度的切分。这样可以确保在查询“语义”或者“切分”和“干预词条”等子短语时能召回改文档。

注意事项

- 用户最多能同时保留8个自定义分词器；
- 单个自定义分词器最多能包含1000个干预词条；
- 每个词条中，key为不超过10个字符，value为不超过32个字符；1个字符为1个汉字或1个英文字母；
- 词条内容中不能包含大写字母(A-Z)，全角符号(\uff01 - \uff5e)，中文标点符号；

语义切分干预词条中，key和value在去掉空格后，内容需相同；示例如下：

```
不正确的词条 => 错误 的 词条
正确的词条 => 正确 的 词条
```

第1个词条中，key和value去掉空格后，内容不一致，因此是不符合规范的词条。

key中不能包含有空格；示例如下：

```
不正确 词条 => 不 正确 词条
正确词条 => 正确 词条
```

第1个词条，key中包含了空格(“ ”)，因此是不符合规范的词条。

key的内容不能为同一个干预词典中其他词条value中的一部分；示例如下：

```
自定义分词器 => 自定义 分词器
```

```
分词器
分词
```

第2个词条的key的内容“分词器”，是第1个词条value中的一部分，因此第2个词条是不符合规范的。但第3条词条是符合规范的

流程演示

流程简述

创建自定义分词器——修改应用结构(分词方式更改为自定义分词器)——索引重建生效自定义分词器效果

具体流程如下：

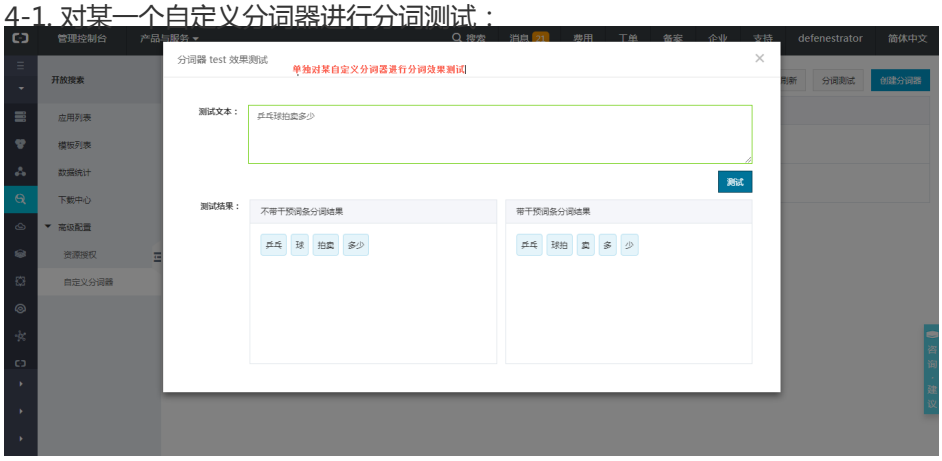
1.在应用列表左侧，找到“高级配置”，选择“自定义分词器”，点击“创建分词器”进行规则添加：

2.创建分词器，按截图中的格式，自定义分词器的名称（示例中创建为了”test”），以及配置需要干预的文档内容以及需要拆分的term（可以配置多条），点击“创建”：

3.创建好名称为“test”的自定义分词器后，等待分词器生效：

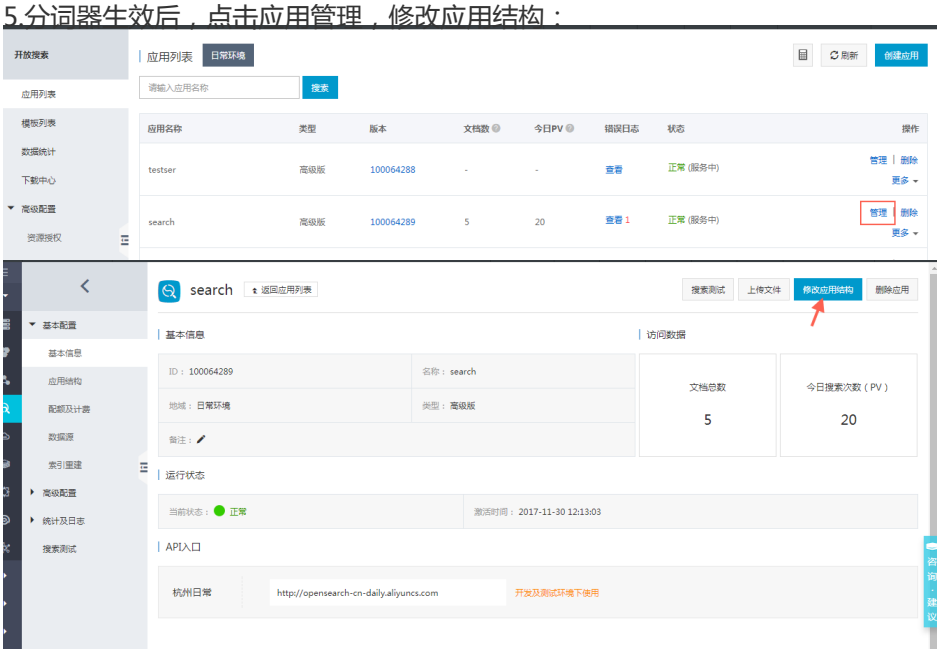
4.分词器生效后，可以进行分词测试，测试分词器生效后干预词条的结果：

- 4-1. 对某一个自定义分词器进行分词测试：



- 4-2. 对自定义分词器列表中的全部或部分进行分词测试：

5.分词器生效后，点击应用管理，修改应用结构：

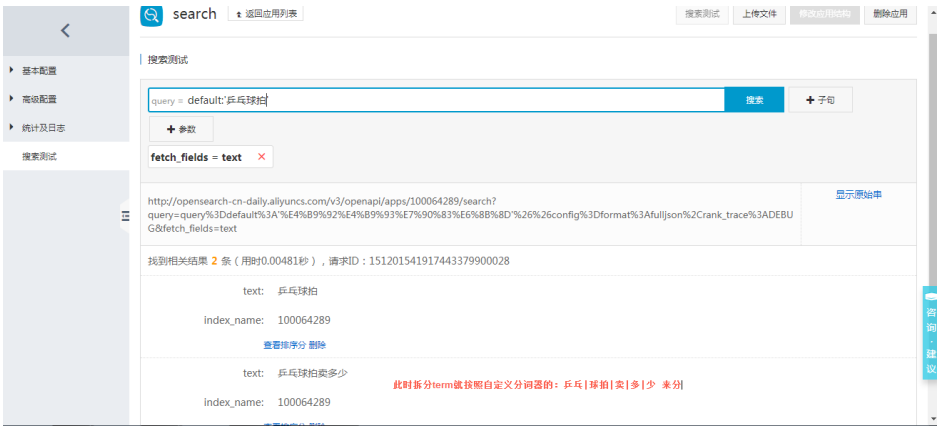


6.在索引字段配置分词方式时，需要指定自定义分词器，索引重建后才会生效该分词器：

自定义分词器效果展示示例：

以“乒乓球拍卖多少”的文档内容为例，当使用“中文——通用分词时”搜索时出现badcase, 与预期不符，如图：

按上述流程添加“test”自定义分词器，并修改应用结构，索引重建后，拆分的term与干预后一致，如图：



注意事项：

1. 自定义分词器不能修改，因此,如果在已经使用test分词器后又发现了搜索的badcase，那么此时需要新建分词器。
2. 自定义分词器有上限，所以如果有应用中不使用的分词器，建议删除。应用中正在使用的分词器是不能删除的。

数据统计

数据统计

说明

目前Opensearch对用户app进行的数据统计指标包括以下6项：

名词	释义
PV	访问OpenSearch搜索接口的次数。
IPV	用户点击OpenSearch搜索结果页中的文档的次数。
UV	访问OpenSearch搜索接口的用户数。同一个用户无论访问多少次都只记1个UV。
IPV_UV	在OpenSearch搜索结果页中点击了文档的用户数。
CTR	点击转化率=IPV/PV

无结果率	OpenSearch返回的搜索结果数(即num)为0的PV/PV。

其中，未开通数据采集服务的app只能展现PV、UV和无结果率。

使用介绍

1. 单日数据默认展现最近一天各个数据指标的情况。可通过日期控件选择展示指标的日期。
2. 数据对比以折线图的方式默认展现最近3天各个数据指标的变化。可通过日期控件选择展示指标的时间跨度。可通过点击图表中指标的icon勾选在图表中展示的指标项。

数据采集

说明

为了给客户提供更高质量的搜索效果，opensearch目前支持客户通过server端上传点击数据。

使用介绍

一、 数据采集功能开通

在控制台内应用下的主菜单栏点击“数据采集”，在页面内仔细阅读《用户承诺书》后，勾选开通Server端数据采集服务和阅读《用途承诺书》后点击确认开通即可完成功能的开通。

二、 埋点规范与行为数据上传

说明

用户在开发搜索控制台开通行为采集功能之后，需要通过api或者sdk手动上传行为数据。本文详细介绍了行为数据包含的字段类型与含义，和采集行为数据时的埋点规范。

行为数据包含的字段定义：

字段名	类型	注释
app_version	string	app的应用版本号，默认为空
channel	string	应用分发渠道，默认为空
imei	string	IMEI（International Mobile Equipment Identity）是国际

		移动设备标识的缩写，IMEI由15位数字(英文字母)组成，默认为空
imsi	string	国际移动用户识别码（IMSI：International Mobile Subscriber Identification Number）是区别移动用户的标志，储存在SIM卡中，可用于区别移动用户的有效信息，默认为空
brand	string	手机或终端的品牌如，TCL，Apple，xiaomi等，默认为空
device_model	string	手机或终端的机型 如：XT800(摩托罗拉)，默认为空
resolution	string	手机或终端的屏幕分辨率 如:800*480(无论是宽还是高,必须大的放前面)，默认为空
os	string	操作系统如: Android、iPhone OS，默认为空
os_version	string	操作系统的版本，默认为空
carrier	string	移动运营商如：中国移动、中国联通、中国电信，默认为空
access	string	连接的网络如：2G、3G、Wi-Fi，默认为空
access_subtype	string	网络类型如：HSPA、EVDO、EDGE、GPRS等，默认为空
client_ip	string	客户端ip地址如，112.96.109.112 可以用来解析国家、省份、城市等信息，不能为空
session_id	string	用户的一次会话id，默认设置: 1>UTDID_TIMESTAMP 2>imei_TIMESTAMP 3>CLIENTIP_TIMESTAMP
reach_time	bigint	到达服务器的时间，格式为：yyyyMMddHHmmss，上传数据时设置
event_id	bigint	埋点的事件ID，详见埋点规范，调用sdk上传数据时sdk自动设置
page	string	当前页面，不能为空
arg1	string	事件参数，默认为空
arg2	string	事件参数，默认为空
arg3	string	事件参数，默认为空

args	string	事件参数，默认为空
local_timestamp	string	终端时间(格式为数字型的unix时间,精确到毫秒,可通过from_unixtime函数转换成日期)，默认为空
utdid	string	服务端生成的设备唯一标识符，默认为空
user_nick	string	长登录会员名称，长登录是指只要登录一次就会记住该设备最近一次登录会员，即使该设备下一次打开App且没有登录，其日志也会记录该设备最近一次登录会员，默认为空
user_id	string	长登录会员id，默认为空
language	string	用户手机端设置语言类型，默认为空
sdk_type	string	sdk类型， 如果使用sdk上传则为'opensearch_sdk'，默认为空
sdk_version	string	opensearch_sdk版本号
reserve1	string	预留字段
reserve2	string	预留字段
reserve3	string	预留字段
reserve4	string	预留字段
reserve5	string	预留字段
reserves	string	预留字段

埋点规范

搜索结果点击埋点

埋点时机：

终端用户在从搜索列表页点击某个搜索结果进入或者退出详情页时，建议在退出时候埋点，这样可以获取用户在详情页停留的时间。

参数定义：

- event_id: 2001
- page：搜索文档的详情页面名称，不能为空
- arg1：搜索结果列表所在的页面名称，不能为空
- arg3：用户在详情页停留的时长（单位ms）

- args：object_id=\$doc_pk,object_type=ops_search_doc,ops_request_misc=xxx

参数说明：

- object_id为被点击的对象的主键，不能为空，如果通过api上传需要urlencod，sdk会自动urlencode。
- object_type必须为ops_search_doc
- ops_request_misc为开放搜索在搜索结果中返回的参数，只要原样设置即可。

行为数据推送

推荐使用opensearch sdk推送行为数据，具体请参考：[aliyun-sdk-opensearch-3.3.0]（java版）、[opensearch-sdk-php-release-v3.1.0]（php版）。

V3.3.0 SDK Commit 方式推送点击行为数据样例代码

Commit 提交数据方式，主要是在程序中动态将对应的点击行为数据封装到Json对象中，再将这些Json对象通过addSearchDocClickRecord 方法添加到缓存中，最后调用Commit方法，批量提交这些点击行为数据。

适用场景

- 动态拼接点击行为数据提交场景
- 单个点击行为数据提交场景
- 小批量性点击行为数据提交场景

java sdk demo:

```
package com.aliyun.opensearch;

import com.aliyun.opensearch.BehaviorCollectionClient;
import com.aliyun.opensearch.OpenSearchClient;
import com.aliyun.opensearch.sdk.generated.OpenSearch;
import com.aliyun.opensearch.sdk.generated.commons.OpenSearchResult;

import java.util.HashMap;
import java.util.Map;

public class PushBehaviorCollectionDoc {
private static String accesskey = "替换accesskey";
private static String secret = "替换secret";
private static String host = "替换应用的API访问地址";

private static String searchAppName = "替换为opensearch应用名";
private static String behaviorCollectionName = "替换为数据采集名称";

public static void main(String[] args) {
//搜索结果列表所在的页面名称
String searchDocListPage = "search_doc_list_page_name";
```

```
//某个搜索文档被点击后，搜索文档的详情页面名称
String docDetailPage = "doc_deatil_page_name";

//用户在详情页停留的时长(单位为ms)
Integer detailPageStayTime = 1500;

//被点击的文档的主键，不能为空
String objectId = "record_pk";

//opensearch返回的查询结果中的ops_request_misc字段
String opsRequestMisc =
"object_id=record_pk,object_type=ops_search_doc,ops_request_misc={\"request_id\":\"153432217417441333635673\", \"scm\":\"a.b.c.d\"}";

//basicFields 为其他基础字段，非必传
//字段介绍请参考
https://help.aliyun.com/document_detail/89971.html?spm=a2c4g.11174283.6.582.b1db5a191ZQxaj
Map<String, String> basicFields = new HashMap<String, String>();
basicFields.put("session_id", "100.69.208.811534340023");
basicFields.put("client_ip", "100.69.208.81");
basicFields.put("reach_time", "1534340023");

//创建并构造OpenSearch对象
OpenSearch opensearch = new OpenSearch(accesskey, secret, host);

//创建OpenSearchClient对象，并以OpenSearch对象作为构造参数
OpenSearchClient client = new OpenSearchClient(opensearch);

//创建BehaviorCollectionClient对象，并以OpenSearchClient对象作为构造参数
BehaviorCollectionClient bcc = new BehaviorCollectionClient(client);

//增加一条搜索点击文档
//这条文档只是增加到sdk client buffer中，没有正式提交到服务端；只有调用了commit方法才会被提交到服务端。
//可以多次调用addSearchDocClickRecord，然后调用commit() 统一提交。
bcc.addSearchDocClickRecord(
searchDocListPage,
docDetailPage,
detailPageStayTime,
objectId,
opsRequestMisc,
basicFields
);

OpenSearchResult openSearchResult;

try {
openSearchResult = bcc.commit(searchAppName, behaviorCollectionName);
System.out.println(openSearchResult);
} catch (Exception e) {
e.printStackTrace();
assertTrue(false);
return;
}
}
}
```

php sdk demo: 创建config配置头文件

```
<?php
//引入头文件
require_once("../OpenSearch/Autoloader/Autoloader.php");
use OpenSearch\Client\OpenSearchClient;
//替换对应的access key id
$accessKeyId = '<Your accessKeyId>';
//替换对应的access secret
$secret = '<Your secret>';
//替换为对应区域api访问地址，可参考应用控制台,基本信息中api地址
$endPoint = '<region endPoint>';
//替换为应用名
$appName = '<app name>';
//替换为数据采集名称
$behaviorCollectionName = '<behavior collection name>';
//开启调试模式
$options = array('debug' => true);
//创建OpenSearchClient客户端对象
$client = new OpenSearchClient($accessKeyId, $secret, $endPoint, $options);
```

php sdk demo: 推送点击行为数据代码

```
<?php
require_once("Config.inc.php");

use OpenSearch\Client\BehaviorCollectionClient;

//搜索结果列表所在的页面名称, 不能为空
$searchDocListPage = "search_doc_list_page_name";

//某个搜索文档被点击后，搜索文档的详情页面名称, 不能为空
$docDetailPage = "doc_detail_page_name";

//用户在详情页停留的时长(单位为ms), 不能为空
$detailPageStayTime = 1000;

//被点击的文档的主键，不能为空
$objectId = "record_pk_name";

//opensearch返回的查询结果中的ops_request_misc字段, 不能为空
$opsRequestMisc = "{\"request_id\":\"153432217417441333635673\", \"scm\":\"a.b.c.d\"}";

//basicFields 为其他基础字段，非必传
//字段介绍请参考
https://help.aliyun.com/document_detail/89971.html?spm=a2c4g.11174283.6.582.b1db5a191ZQxaj
$basicFields = array();
$basicFields['client_ip'] = "100.69.208.81";
$basicFields['reach_time'] = "1534340023";

//增加一条搜索点击文档
//这条文档只是增加到sdk client buffer中，没有正式提交到服务端；只有调用了commit方法才会被提交到服务端。
//可以多次调用addSearchDocClickRecord，然后调用commit() 统一提交。
$bcc = new BehaviorCollectionClient($client);
```

```
$bcc->addSearchDocClickRecord(
$searchDocListPage,
$docDetailPage,
$detailPageStayTime,
$objectId,
$opsRequestMisc,
$basicFields
);

$ret = $behaviorCollectionClient->commit($appName, $behaviorCollectionName);

print_r(json_decode($ret->result, true));
```

三、开通后的使用场景

服务开通后采集到的数据目前可在类目预测训练模型中选择使用，同时默认在App数据概况、A/B Test数据统计报表内使用。采集到的数据Opensearch仅会保留最近10天的数据。

四、关闭数据采集服务

如果不再希望Opensearch采集并使用数据，可在控制台数据采集功能页面内关闭服务。关闭后Opensearch会立即停止数据的采集同时不再保留所有的历史上传数据。

类目预测

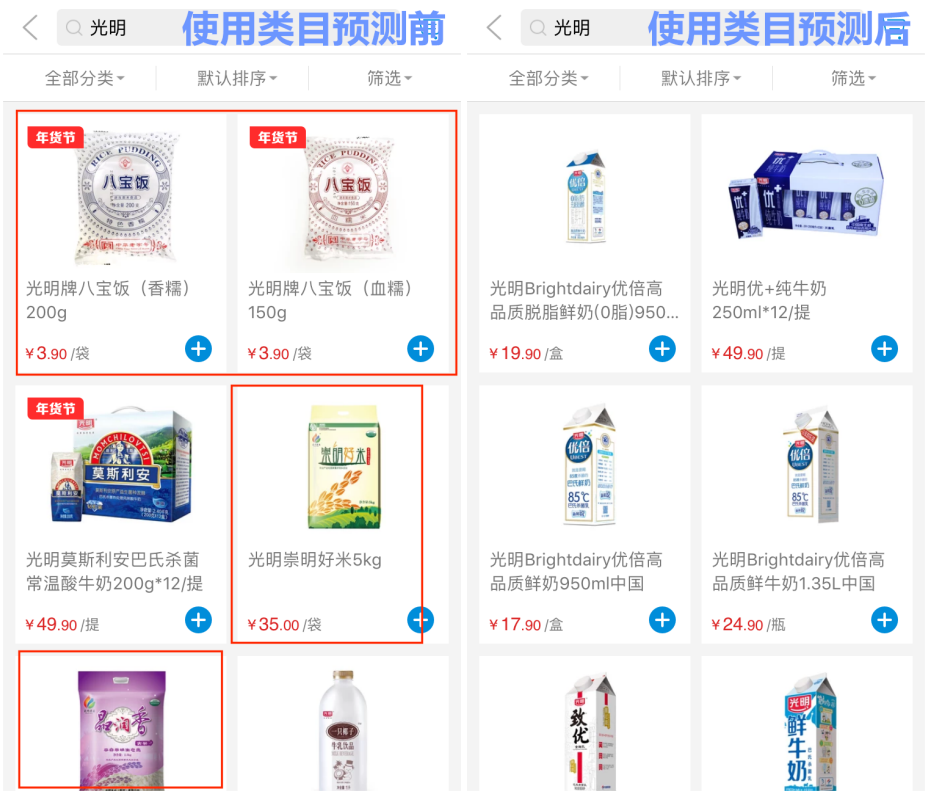
基本概念

类目预测

根据用户的查询字符串来预测用户是想要查询那个类目的结果，比如当用户搜索“苹果”时，类目预测可能的预测结果为：

- 电子产品类目的可能性为2。
- 水果类目的可能性为1。

类目预测的结果会作用于搜索结果排序，使得更符合用户搜索意图的结果更加靠前。如下图所示，在电商场景下，对“光明”的搜索Query进行类目预测后的结果：



模型：

本语境下，特指基于用户数据训练出来的类目预测数据结果，在线搜索时直接查询该结果。

使用前提

有阿里云账号并且有正在使用的新高级版Opensearch应用（**标准版和老高级版，不支持类目预测功能**）。

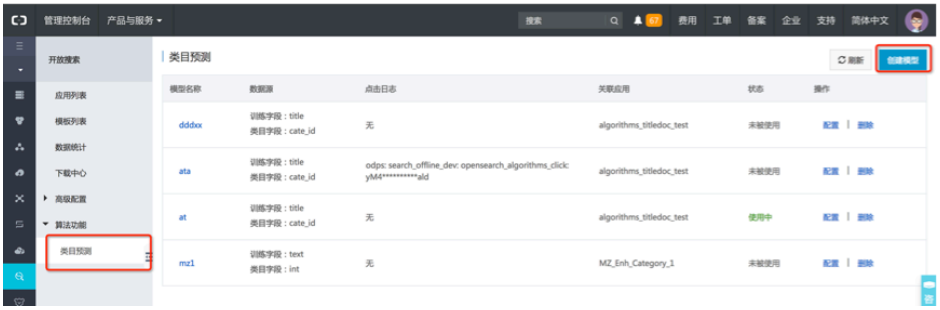
Opensearch应用主表中有定义**训练字段**和**类目字段**，其中**训练字段**应为Text类型，**类目字段**应为INT类型。

我们支持有点击日志的训练和没有点击日志的训练，所以有无点击日志均可以使用类目预测。

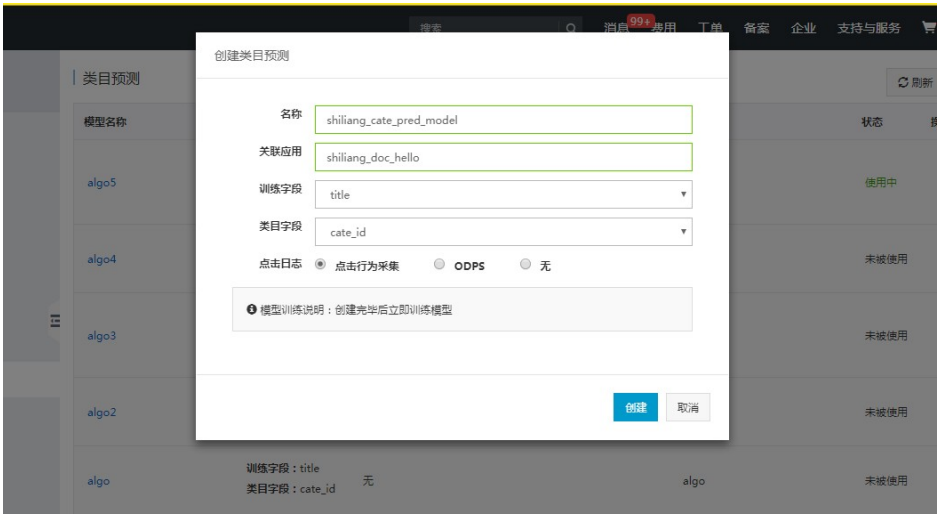
操作步骤

创建模型

打开控制台，点击左侧“算法功能”内“类目预测”按钮，然后点击右上角“创建模型”按钮：



填写类目预测模型名称、关联应用（必须为新高级版应用）、训练字段和类目字段（必须为INT）。第一次关联的应用使用“点击行为采集”需要先开通“数据采集”服务，具体可以查看“数据采集”文档。如果有点击日志则需提供点击日志所在的ODPS项目、表和Access ID：



所有字段填写完成后，点击创建。

查看模型运行状态

创建成功后，自动跳转到模型列表页，可以看到刚刚创建的模型：

点击模型名称或者配置按钮，进入模型详情页：

模型训练通常会花费小时级别时间，可以在模型详情页查看训练进度。

效果测试

待模型训练完成生效后，可以点击模型详情页右上角“搜索测试”按钮，输入搜索字符串，查看该模型对该搜索字符串所预测的各类目得分：

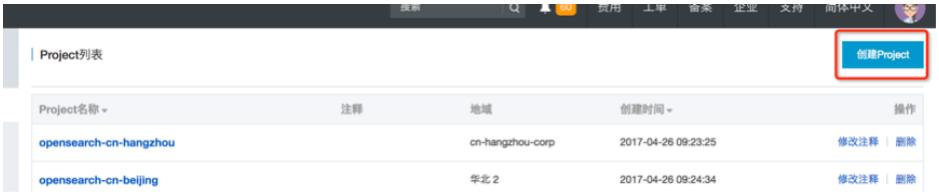
如上图，“苹果”对应的类目1得分为2，类目2的得分为2，类目5得分为1。

类目预测结果将作用于该搜索串的搜索结果排序，相比未应用类目预测的结果，应用类目预测后的搜索结果中，类目1和类目2的结果可能会相对更加靠前。

使用SLS同步clicklog步骤

创建SLS项目

打开地址：https://sls.console.aliyun.com/#/

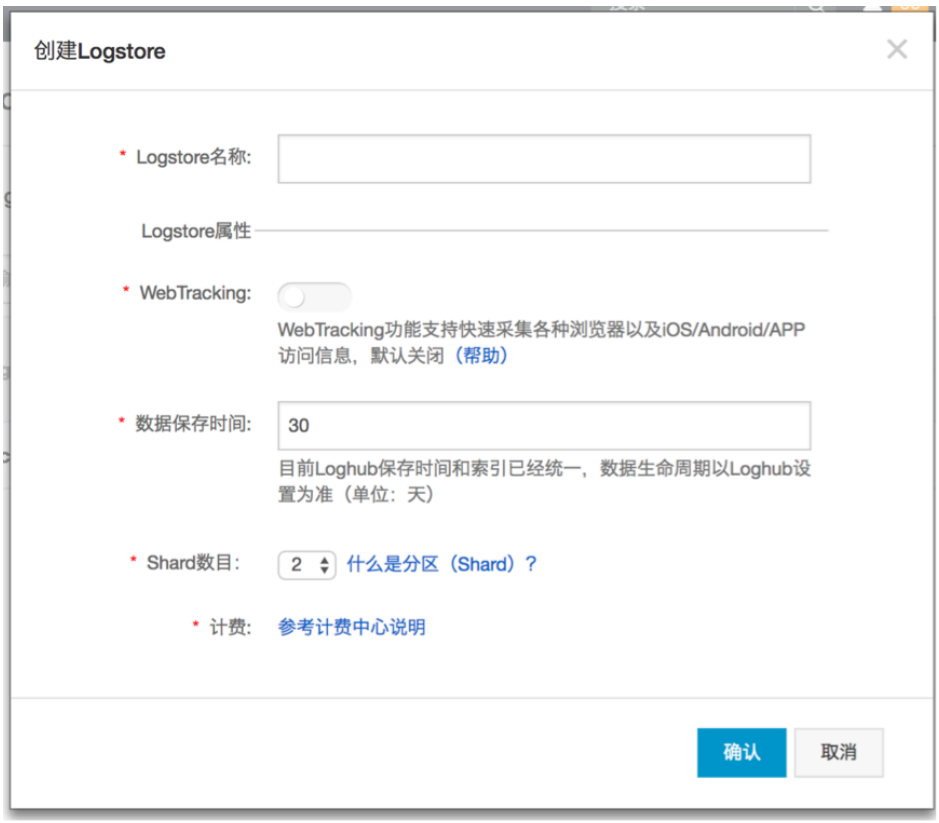


进入控制台后，创建SLS项目：

填写Project名称并选择区域（应该和Click log所在ECS区域相同），创建成功后会出现以下提示：

创建Logstore

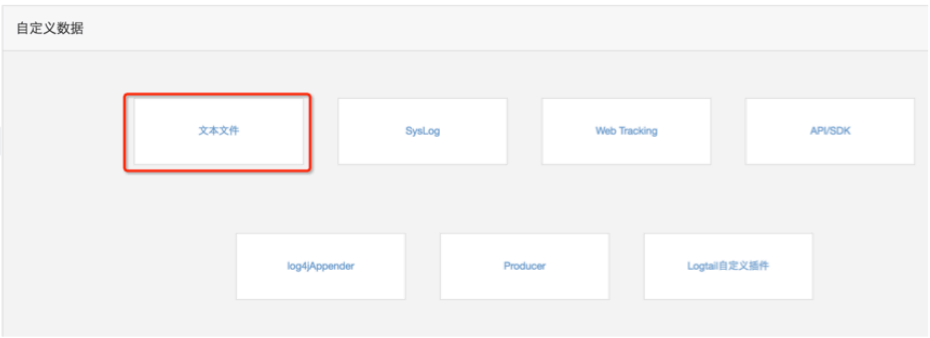
点击上图中创建出现下图：



填写Logstore名称，其余保持默认。创建log store之后，出现以下提示：

创建数据接入配置

将页面拉到最下方，点击“文本文件”，然后点击下一步，进入数据源设置页面：



设置配置名称和日志路径。

比如收集/root/shiliang/click.log.json的数据，那么就在日志路径行第一个空白处填写/root/shiliang，第二个空白处填写click.log.json：

模式选择为使用JSON模式，然后保持高级选项不变，点击下一步：

创建机器组并应用数据接入配置

点击“创建机器组”创建机器组（如果已有，应用到已有机器组即可）：

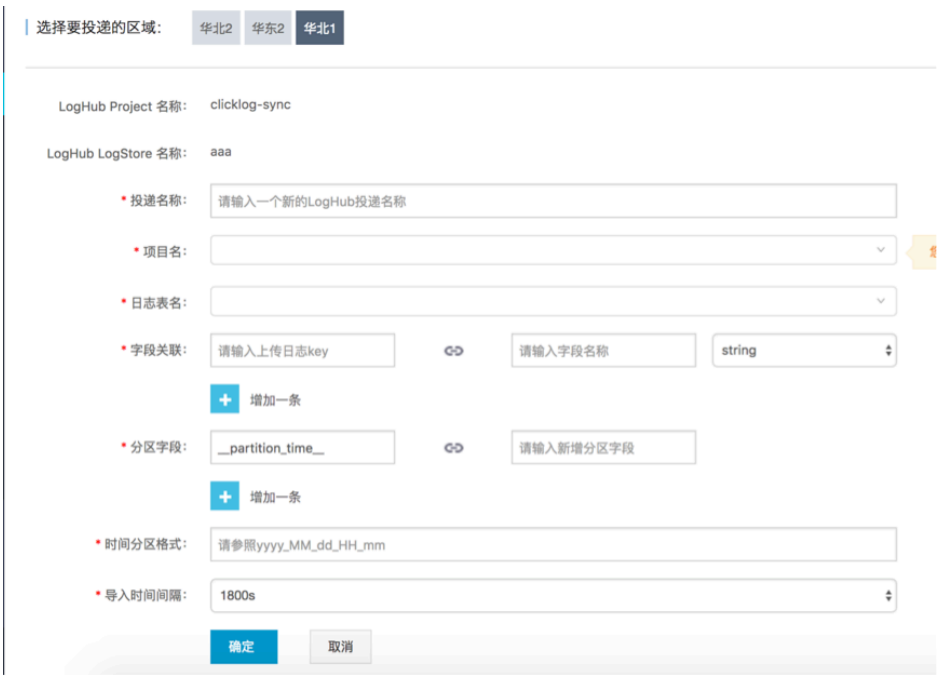
填写机器组的名称和ECS VM IP地址即可（填写内部IP，非公网IP）：

勾选刚刚创建的机器组，然后点击“应用到机器组”：

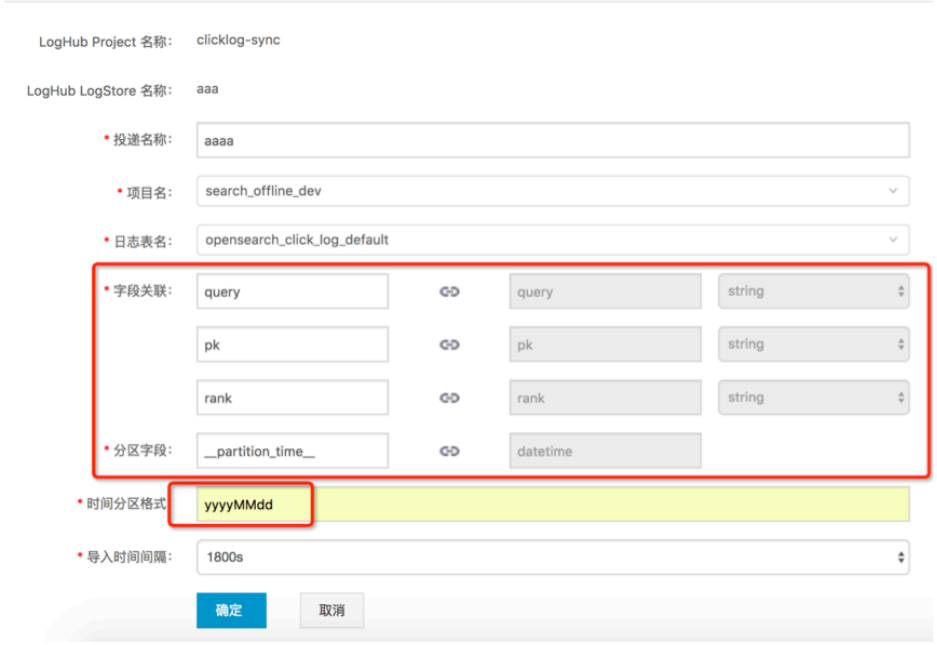
在查询分析&可视化步骤不做操作，点击下一步：

在“投递MaxCompute”栏点击“开启投递”，会跳转到数加控制台。

开启MaxCompute投递



选择投递区域，设定投递名称，选择要投递到的项目名和日志表名。选择后，页面右侧会出现ODPS表的各个native字段和partition字段（如下）：

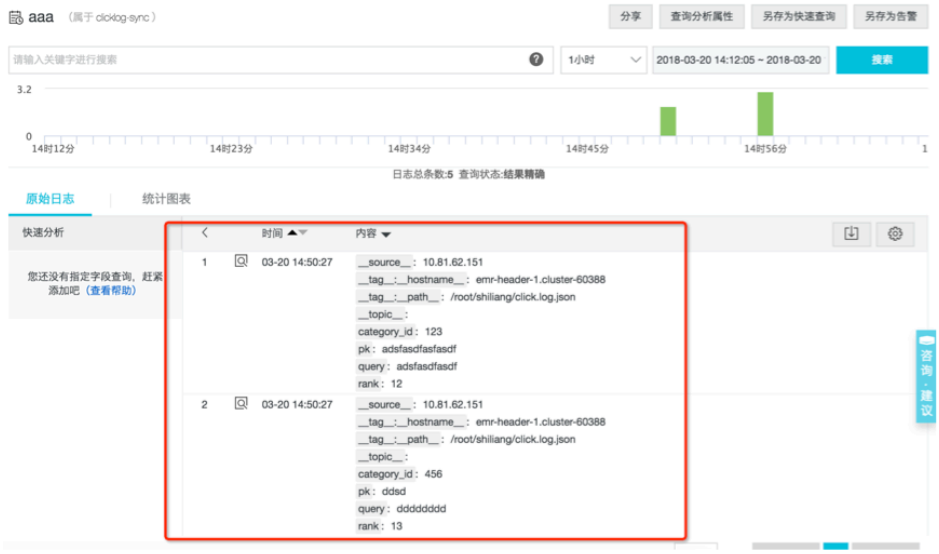


分区字段使用**partition_time**，时间分区格式使用yyyyMMdd，其余配置保持不变即可。

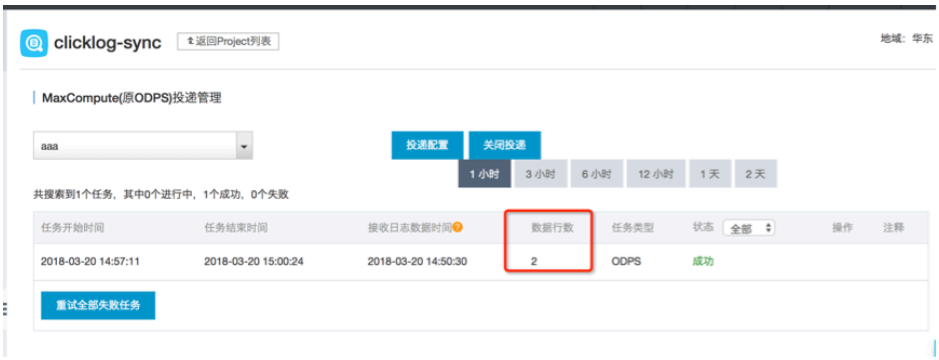
检查日志同步状态

在logstore列表中选择自己的logstore，点击查询，可以查看已经收集到SLS的日志。点击MaxCompute可以查看每次投递任务的状态:

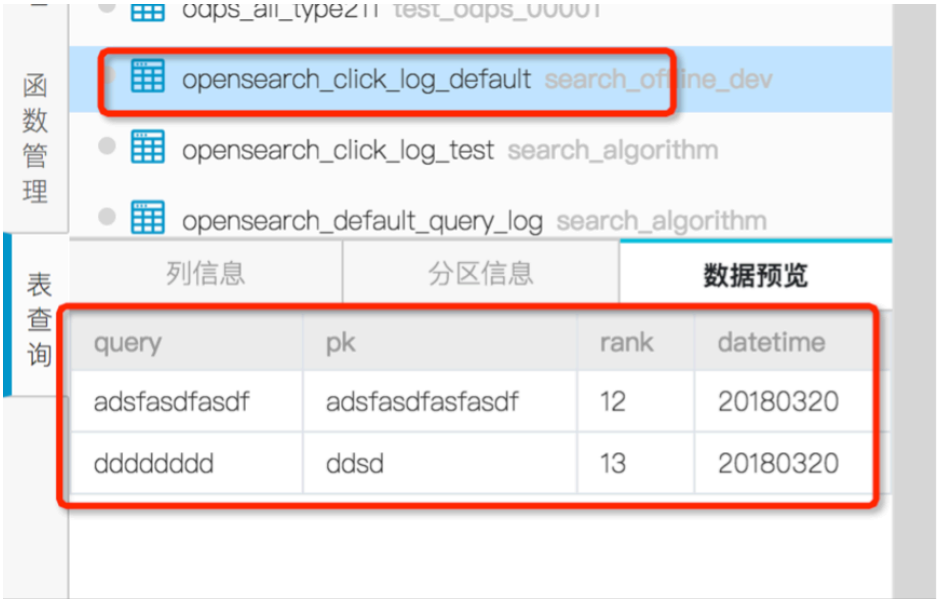
已经收集到SLS的2条数据：



2条数据已经都投递到了MaxCompute：



在MaxCompute的web IDE中可以看到：



在日志成功投递到MaxCompute之后，就可以使用其进行下一步计算。

应用类目预测模型

类目预测功能的目的是为了提升搜索结果的排序效果。例如预测到用户输入“苹果”是想要搜“电子产品”类的苹果手机，而不是“水果”类的可以吃的苹果，那么搜索结果中，苹果手机的搜索结果就应该更靠前。

类目预测模型的应用，顾名思义是将该类目预测模型配置到开放搜索应用里去，再修改粗排或精排表达式，将类目预测算分特征函数category_score()添加到表达式中，最后搜索时配置category_prediction参数，即可观察到搜索提升后的效果。

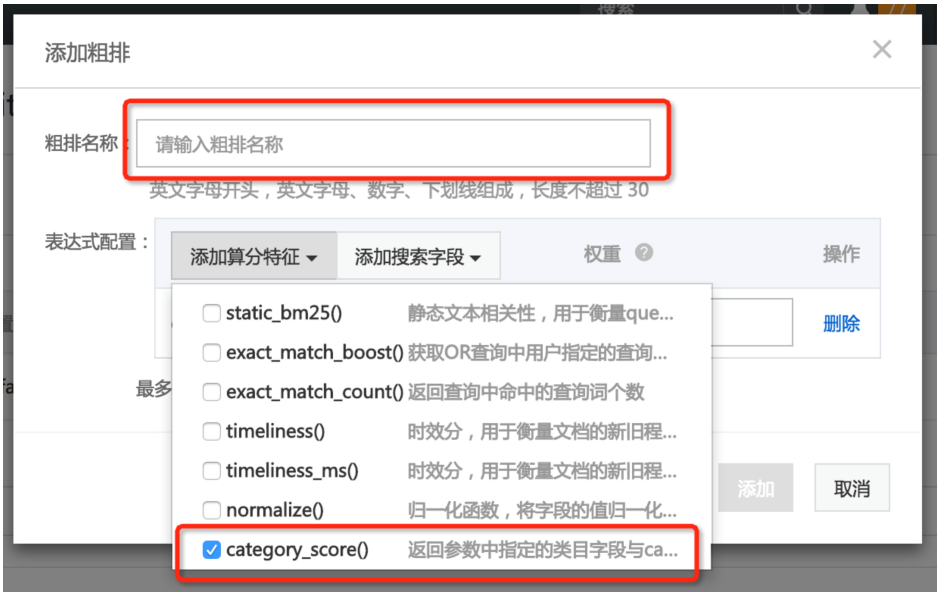
关于排序表达式使用，详情请参见搜索相关性配置。

应用类目预测到排序

点击控制台左侧 “高级配置”下的“搜索结果排序” ，进入搜索结果排序配置页面，然后点击右上角“修改表达式” ：

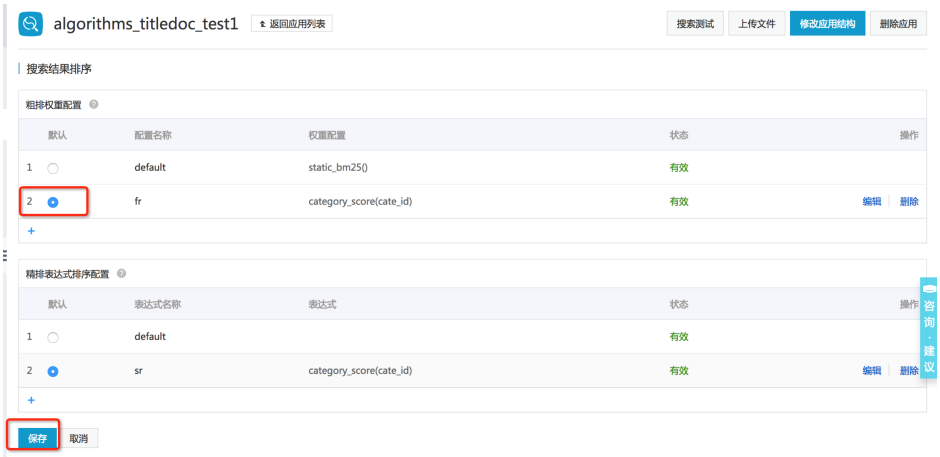
点击后，可以编辑现有的排序表达式或者增加新的排序表达式：

点击粗排表达式下的加号，增加一条新的排序，如下图，输入粗排名称，然后选择category_score()算分特征：



然后指定一个字段（**只能是用于训练时作为类目id的字段**）作为category_score()的输入参数计算类目得分，再指定权重，点击添加：

添加后，选中刚刚新建的排序表达式，然后点击保存按钮，即可生效:



类目预测既可以应用在粗排表达式中（如本例所示），也可以应用于精排表达式中，配置方式相同。

应用类目预测到搜索

在搜索时需要添加category_prediction参数，才能将指定类目预测应用到搜索。

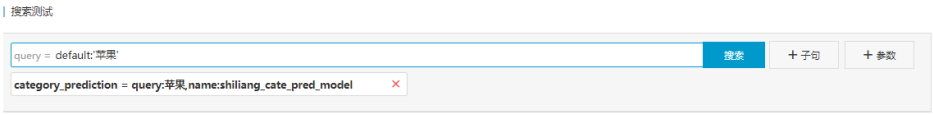
参数说明

category_prediction=query:请填写查询词,name:请填写类目预测名称

- **query**：必选，用于类目预测的query查询。
- **name**：可选，不指定则使用默认生效的类目模型。

搜索测试

在开放搜索控制台搜索测试界面中进行搜索如下图所示。



温馨提示：

通过SDK调用时，也需要指定category_prediction参数，才能应用类目预测到搜索。

常见问题

如果模型跑的时间过长或者模型训练结果为失败，请在模型详情页点击**“查看记录”**按钮，然后查看有问题的任务ID，提工单描述应用信息以及该任务ID，以便我们查找问题。

A/B Test

说明

为了让用户更加合理地使用、调试OpenSearch的各种算法，上线A/B Test功能，方便业务在全量使用前可以分配一定比例的流量进行先验，避免盲用带来对线上业务的负面影响。A/B Test功能目前支持用户对查询分析、粗排表达式、精排表达式以及类目预测进行实验配置。

使用介绍

一. 名词解释

名词	释义
A/B Test	A/B Test是对同一个算法功能制作两个（A/B）或多个（A/B/n）版本，在同一时间维度，分别让组成成分相同（相似）的访客群组随机的访问这些版本，收集各群组的用户体验数据和业务数据，最后分析评估出最好版本正式采用。
实验场景	指的是准备进行实验的全部流量集合。
场景标识	应用内不同的流量集合标记，根据用户自身业务来标记区分；对应查询参数abtest中scene_tag。
实验组	实验的容器，目前一个实验场景下只能创建一个实验组。
实验	实验组内用来测试算法功能的随机流量。
实验分流	实验流量随机切分的方式，对应查询参数abtest中flow_divider。
PV	访问OpenSearch搜索接口的次数。
IPV	用户点击OpenSearch搜索结果页中的文档的次数。
UV	访问OpenSearch搜索接口的用户数。同一个用户无论访问多少次都只记1个UV。
IPV_UV	在OpenSearch搜索结果页中点击了文档的用户数。
CTR	点击转化率=IPV/PV
无结果率	OpenSearch返回的搜索结果数(即num)为0的PV/PV。

实验场景、实验组和实验的逻辑关系

二. 配置A/B Test的基本流程

为了最终的实验指标更全面，在配置A/B Test之前建议先开通点击行为采集数据。用户初次使用A/B Test功能需要经过以下四个步骤才能完成具体实验的配置上线：

进入A/B Test创建流程

首次进入点击“立即创建”即可进入A/B Test创建流程。

创建实验场景

实验场景的创建是为了明确应用中哪一部分流量集合是用户准备做实验的。每个应用下最多只能创建3个实验场景。

2.1 场景标识

创建时用户首先需要填写场景标识，场景标识是实验场景识别的唯一标志，同一个实验场景可以由多个场景组成，所以场景标识可以填写多个。场景标识必须是字母数字下划线组合，不超过 30 个字符，非 ops_ 开头。场景标识最多 10 个。如果不需要实验场景划分，场景标识不填即可，代表该实例下的全部流量均可被实验。要注意的是，由于每个实验场景所包含的场景标识不可以有交叉，所以当一个实验场景内包含所有场景标识，就不能再新建另一个场景。

2.2 实验场景名称

实验场景名称仅做展示使用，默认生成规则为Scene_创建日期_创建时间，用户可修改。场景名称不允许超过 30 个字符。

创建实验组

实验组是隶属于实验场景的，目前一个实验场景内仅支持创建一个实验组，这个实验组将默认绑定所有的实验参数，也就是说目前在实验场景内唯一的实验组下，可以对查询分析、粗排表达式、精排表达式以及类目预测进行实验。

3.1 实验组名称

实验组名称仅做展示使用，默认生成规则为Group_创建日期_创建时间，用户可修改。实验组名称不允许超过 30 个字符。

创建实验

实验组创建好以后点击“添加实验”就可以在实验组内创建具体的实验了。每个实验组里面最多只能创建20个实验，其中最多只能有10个实验同时在线生效。

4.1 实验名称

实验名称必须用户自定义，不超过 30 个字符。

4.2 添加配置

用户可选择对查询分析、粗排表达式、精排表达式以及类目预测进行配置。配置项弹框只支持用户进行选择已有的查询分析规则、粗排表达式、精排表达式以及应用关联的类目预测模型。用户如需新增或者修改当前已有内容，点击“添加或编辑”进入对应配置项的新增/修改页，在新增/修改页上配置完毕点击保存后返回配置项弹窗，此时弹窗里会同步展示出用户新增的相关内容。一个实验可以最多添加4个配置，对已添加的配置，可点击配置生成的方块右上方删除按钮进行删除。

4.3 实验流量

实验流量最小粒度为1%，同一场景的同一实验组内在线的所有实验流量总和应该≤100%。

5 . A/B Test实验生效

控制台开通A/B Test功能，并配置好实验之后，如果想让实验对线上的查询生效，需要在查询中指定abtest参数。abtest参数主要包含2个部分：scene_tag和flow_divider。

- scene_tag：场景标识，如果用户在控制台上没有设置表示对所有场景下的流量都会做实验，查询中也不用设置。如果用户在控制台上设置多个场景标识(用分号分割)，查询中只要设置其中的一个值即可。
- flow_divider：后端系统对该值进行hash将用户的查询流量分配到不同的实验中，分配的比例为用户在控制台上配置的实验流量比例。flow_divider推荐为最终用户的id，如果没有的话可以用最终用户的设备id或者ip地址。

如果通过api访问开放搜索服务，scene_tag和flow_divider的值需要urlencode，最终传给开放搜索的abtest参数格式为abtest=urlencode(scene_tag:urlencode(\$scene),flow_divider:urlencode(\$value))，其中urlencode为url编码函数。如果通过sdk访问开放搜索服务，只需要调用对应的接口即可，scene_tag和flow_divider无需encode处理，详细请参考“六.实战示例”。

三. 实验管理

创建好实验场景、实验组和实验后，会生成对应的实验列表。实验列表内可切换展现其他已创建过的实验场景下的实验列表。

实验列表内展现了实验名称、实验状态、实验配置、实验流量和更新时间等基本信息。用户可对已创建的实验进行基础的管理操作。

编辑实验

支持对实验名称、实验配置、实验流量进行修改。

删除实验

实验被删除后，系统将不再保留其实验配置信息，并且线上不再生效。

下线实验

实验被下线后，实验流量将被切为0，但是会在列表里保留实验的其他配置信息。

启用实验

被下线的实验，通过重新编辑流量后，点击启用即可上线重新生效。

四. 场景管理

场景管理页内是已创建的实验场景列表，可对列表里的实验场景进行编辑和删除。

编辑实验场景

支持对实验场景名称、标识进行修改。

删除实验场景

实验场景删除后，场景下的实验组以及实验组内的所有实现也均被删除，线上将不再生效，且系统不会再保留所有相关的配置信息。请谨慎处理。

五. A/B Test实验数据报表

从A/B Test实验列表页内点击查看实验数据或者直接从应用下的菜单栏内点击数据统计-A/B Test数据统计即可进入实验数据报表页。页面包含核心数据指标对比页和具体指标数据表格两个tab。实验数据报表目前是按照实验场景-实验组粒度展现，用户可以在页面内切换选择其它实验场景查看对应的实验数据。

核心数据指标

默认展现的是最近1天的指标数据。用户可在页面内的实验选择下拉框内勾选多个要对比数据的实验。目前的可统计的核心指标有PV、IPV、UV、IPV_UV、CTR、无结果率，未开通点击数据采集服务的核心指标只支持统计PV、UV和无结果率。

具体数据表格

具体数据页面内展现的是各个实验下指标的天级具体数据以及每个指标在周期内每天的趋势变化。页面默认展现最近3天各个实验的相关数据，用户可自定义修改查看的日期跨度。展现的指标项同核心数据表格。点击表格内的指标旁边的icon即可展现该指标的变化曲线图。

六. 实战示例

某电商产品线上使用OpenSearch查询有以下两种场景：

第一种场景：来自终端客户进行的商品关键词搜索流量，查询方式为：

query=config=format:fulljson&&query=default:'宝宝奶粉'&&sort=price

第二种场景：来自内部其他业务的调用流量，查询方式为：

query=config=format:fulljson&&query=cat_id:'1'|'2'|'3'&&sort=timestamp

现在该用户希望在第一种场景下，按照终端用户会员id分流进行A/B Test，对比几种排序表达式或者类目预测模型或者查询分析规则的效果。那么这个用户的配置步骤如下：

在控制台A/B Test功能里创建场景、实验组和实验。其中在创建场景的步骤中，将场景标识填写为user_search 代表第一种场景。

在查询中设置A/B Test参数。由于在控制台已填写场景标识为 user_search，所以在场景一的查询中应该设置参数scene_tag:user_search,flow_divider:xxxx(终端用户会员id的值)。

2.1 SDK方式(如下是java sdk用法，php sdk用法类似)

[aliyun-sdk-opensearch-3.3.0]（java版）、 [opensearch-sdk-php-release-v3.1.0]（php版）。

2.2 **API方式**

i. query=config=format:fulljson&&query=default:'宝宝奶粉'&&sort=-price&abtest=scene_tag:user_search,flow_divider:%e5%bc%a0%e4%b8%89

> **注意：**
>
> 这里对abtest的子参数scene_tag和flow_divider的value都做了urlencode）。

ii. **对请求中的每个参数(即query, sort, abtest)的value做urlencode:**

query=config%3dformat%3afulljson%26%26query%3ddefault%3a%27%e5%ae%9d%e5%ae%9d%e5%a5%b6%e7%b2%89%27&&sort=-price&abtest=scene_tag%3auser_search%2cflow_divider%3a%25e5%25bc%25a0%25e4%25b8%2589

以上配置完毕后，就可以实现对第一种场景的流量进行A/B Test的需求了。

干预功能

类目预测干预功能

使用介绍

目前支持对训练出的类目预测模型进行人工干预。用户实现干预操作的过程与查询分析干预类似，通常有以下三步：

1. 创建干预词典。用户进入到类目预测干预词典页后，点击页面右上角的“创建词典”。为词典命名后，即可完成词典创建。
2. 新增和管理干预词典内的干预词条。词典创建完成后，在列表中点击词典名称或点击词典对应的“管理”，即可进入到干预词典的详情页。用户可在详情页内进行干预词条的新增和管理。用户可对某个Query干预召回结果中类目的相关度，纠正模型预测相关度错误的类目或补充模型未预测出相关度的类目。
3. 使用干预词典。创建并填充完成类目预测的干预词典后，可在任意的类目预测模型使用。干预词典使用后，不会参与到模型的训练中，如果在实际查询中干预词典中和类目预测模型中都有Query下同一类目的相关度计算结果，那么会使用干预词典内结果进行计算。比如搜索“微微一笑很倾城”，类目预测模型计算出该Query类目20的相关度是0（不相关），但是这个类目预测使用的干预词典中“微微一笑很倾城”的类目20的相关度被干预成了2（相关），那么线上会使用类目20的相关度为2的结果来进行计算。

实战演示

- 业务场景：某电商类业务在OpenSearch的应用实例中配置使用了类目预测模型，但是在线上发现了一些badcase，于是决定使用干预功能。
- **Badcase：**用户搜索Query“牛奶”，返回的结果中牛奶杯的商品靠前，真正的牛奶靠后。
- 问题诊断：类目预测模型将Query“牛奶”的类目相关性计算错误。
- 解决方案：新建类目预测干预词典，在词典中干预Query“牛奶”，将“牛奶杯”所属的家居用品类目（id是20)的相关度定为略相关，将“牛奶”所属的食品类目（id是15）的相关度定为“相关”。
- **配置流程**：

1.点击控制台首页干预功能：类目预测干预词典。

2.创建一个类目预测干预词典，命名为"leimutest"。

3.在"leimutest"里新增干预词条，Query栏填写“牛奶”，类目ID-相关度栏填写“20-1;15-2”（2代表相关，1代表略相关，0代表不相关）。

4.将干预词典应用到线上使用的类目预测模型。

注意事项

1. 词典名称在创建后不可修改。
2. 新增干预词条时，填写的Query不应与干预列表内已干预过的Query重复。历史干预过的Query可在列表内直接对已有的类目ID-相关度进行增、删、改。
3. 单个Query下添加多个类目ID-相关度组合时，每个组合之间用;分隔。
4. 新增或修改干预词条后，生效状态如果持续是“正在生效”，可以点击刷新按钮获取生效状态的同步。
5. 同一类目预测干预词典可以被多个类目预测模型使用。
6. 干预词典不会影响参与模型的训练。
7. 被任一模型（不论是否上线）使用的干预词典不能被删除，想要删除需要首先解除使用。

功能限制

1. 类目预测干预词典一共可以创建10个。
2. 每次新增干预词条时，Query只支持填写一个，单个Query下最多添加5个类目ID-相关度的组合。
3. 每个类目预测干预词典最多可创建500个干预词条。
4. 类目预测干预的Query与查询Query精准匹配才生效。例如，Query“连衣裙”干预了“类目23相关度为相关”，那么查询“连衣裙”时类目23的结果会按照相关结果进行排序分计算，但是查询“2018连衣裙”时，类目23不会使用上述干预内容进行计算。
5. 添加的干预内容均会进行大小写和全半角归一化处理，其中大写字母会归一化为小写，全角会归一化为半角。

查询分析-同义词干预功能

使用介绍

目前支持对系统内置的同义词词典进行人工干预。用户实现干预操作的过程通常有以下四步：

1. 创建同义词干预词典。用户进入到查询分析干预词典页后，点击页面右上角的“创建词典”。选择了词典类型后，为词典命名，干预词典创建完成，词典会出现在页面的词典列表中。
2. 新增和管理干预词典内的干预词条。词典创建完成后，在列表中点击词典名称或点击词典对应的“管理”，即可进入到干预词典的详情页。用户可在详情页内进行干预词条的新增和管理。用户可对Query进行两种类型的干预，-添加同义词：对一个Query添加同义词，系统在查询Query时会同时返回包含Query的或添加的同义词的搜索结果。-屏蔽同义词：对一个Query屏蔽同义词，系统在查询Query时不再会同时返回包含屏蔽的同义词的搜索结果。
3. 使用干预词典。创建并填充完成同义词干预词典后，可在任意应用的查询规则内选择使用。
4. 干预词典效果测试和上线。查询分析规则使用了干预词典后，在应用到线上前建议先进行搜索效果测试，评估效果是否符合干预预期。

实战演示

- 业务场景：某电商导购类业务在OpenSearch的应用实例中配置使用了查询分析规则，规则包含同义词功能，但是在线上发现了badcase，于是决定使用干预功能。
- **Badcase：**用户搜索Query“契尔氏”发现相关商品没有返回，返回的结果里均包含“屈臣氏”，但实际数据库内有相关商品的数据，不过商品的描述都用了其同义词“科颜氏”。
- 问题诊断：a. 系统同义词词典错误的把“屈臣氏”当作“契尔氏”的同义词进行了召回；b. 系统同义词词典缺失”科颜氏“是”契尔氏“的同义词识别。
- 解决方案：新建同义词干预词典，在词典中干预Query“契尔氏”，为其添加同义词“科颜氏”，屏蔽错误识别的同义词“屈臣氏”。将该同义词干预词典应用在线上使用的查询分析规则中。
- 配置流程：1.点击控制台首页干预功能：查询分析干预词典。

2.创建一个同义词干预词典，命名为”tongyicitest”。

3.在”tongyicitest”里新增干预词条，Query栏填写“契尔氏”，添加同义词栏填写“科颜氏”，屏蔽同义词栏填写“屈臣氏”。

4.在app中把干预词典先应用在一个未上线的查询分析规则中，以便进行搜索效果测试。

5.搜索测试是否符合预期效果。发现搜索结果召回了包含“科颜氏”或“契尔氏”的结果，没有再召回包含“屈臣氏”的结果，符合预期。

6.将干预词典应用到已上线的查询分析规则中。

注意事项

1. 词典类型和名称在创建后均不可修改。
2. 新增干预词条时，填写的Query不应与干预列表内已干预过的Query重复。历史干预过的Query可在列表内直接对增加和屏蔽的同义词进行增、删、改。
3. 单个Query下添加或屏蔽多个同义词，每个词之间用;分隔。
4. 新增或修改干预词条后，生效状态如果持续是“正在生效”，可以点击刷新按钮获取生效状态的同步。
5. 同一干预词典可以被多个查询分析规则使用。
6. 由于目前的干预词典是基于系统内置词典的补丁式干预，所以词典使用时会默认勾选使用系统内置词典。
7. 被任一查询规则（不论是否上线）使用的干预词典不能被删除，想要删除需要首先解除使用。

功能限制

1. 同义词干预词典一共可以创建10个。
2. 每次新增同义词干预词条时，Query只支持填写一个，同一Query下添加和屏蔽的同义词总和应≤5条。
3. 每个同义词干预词典最多可创建500个干预词条。
4. 同义词干预词条按照分词后的term进行匹配生效。例如，为Query“北京”添加了同义词“帝都

”，那么查询“北京”会返回包含“北京”或“帝都”的结果，同时查询“北京欢迎你”，返回包含“北京欢迎你”或“帝都欢迎你”的结果。

5. 添加的干预内容均会进行大小写和全半角归一化处理，其中大写字母会归一化为小写，全角会归一化为半角。

查询分析-停用词干预功能

使用介绍

目前支持对系统内置的停用词词典进行人工干预。用户实现干预操作的过程通常有以下四步：

1. 创建停用词干预词典。用户进入到查询分析干预词典页后，点击页面右上角的“创建词典”。选择了词典类型后，为词典命名，干预词典创建完成，词典会出现在页面的词典列表中。
2. 新增和管理干预词典内的干预词条。词典创建完成后，在列表中点击词典名称或点击词典对应的“管理”，即可进入到干预词典的详情页。用户可在详情页内进行干预词条的新增和管理。用户可进行两种类型的干预，-添加停用词：添加一个停用词后，如果查询的Query中分词后有term为添加的这个停用词，在召回时该term将不参与召回。-屏蔽停用词：屏蔽一个停用词后，如果查询的Query中分词后有term为屏蔽的这个停用词，在召回时该term将正常参与召回。
3. 使用干预词典。创建并填充完成停用词干预词典后，可在任意应用的查询规则内选择使用。
4. 干预词典效果测试和上线。查询分析规则使用了干预词典后，应用到线上之前建议先进行搜索效果测试，评估效果是否符合干预预期。

实战演示

- 业务场景：某电商导购类业务在OpenSearch的应用实例中配置使用了查询分析规则，规则包含停用词功能，但是在线上发现了badcase，于是决定使用干预功能。
- **Badcase：**用户搜索Query“什么面霜好呢”，返回的结果寥寥无几，但是都完整包含了“什么面霜好呢”关键词，实际上数据库内还有很多语义相关的结果没有召回。
- 问题诊断：原因之一是Query中的“呢”没有被系统识别成停用词。
- 解决方案：新建停用词干预词典，在词典中把“呢”添加为停用词，再将该停用词干预词典应用在线上使用的查询分析规则中。
- 配置流程：1.点击控制台首页干预功能：查询分析干预词典。

2.创建一个停用词干预词典，命名为"tingyongcitest"。

3.在"tingyongcitest"里新增干预词条，停用词栏填“呢”，干预类型选择“添加”。

4.在app中把干预词典先应用在一个未上线的查询分析规则中，以便进行搜索效果测试。

5.搜索测试是否符合预期效果。发现搜索结果召回了不包含“呢”但是仍然需求相关的结果。

注意事项

1. 词典类型和名称在创建后均不可修改。
2. 新增干预词条时，填写的停用词不应与干预列表内已干预过的停用词重复。
3. 新增或修改干预词条后，生效状态如果持续是“正在生效”，可以点击刷新按钮获取生效状态的同步。
4. 同一干预词典可以被多个查询分析规则使用。
5. 由于目前的干预词典是基于系统内置词典的补丁式干预，所以词典使用时会默认勾选使用系统内置词典。
6. 被任一查询规则（不论是否上线）使用的干预词典不能被删除，想要删除需要首先解除使用。

功能限制

1. 停用词干预词典一共可以创建10个。
2. 每次新增停用词干预词条时，只支持填写一个停用词。
3. 每个停用词词典最多创建500个干预词条。
4. 停用词干预词条按照分词后的term进行匹配生效。例如，将“吧”干预添加成为停用词，那么查询“一起玩游戏吧”时“吧”不会参与召回。
5. 添加的干预内容均会进行大小写和全半角归一化处理，其中大写字母会归一化为小写，全角会归一化为半角。

查询分析-拼写纠错干预功能

使用介绍

目前支持对系统内置的拼写纠错词典进行人工干预。用户实现干预操作的过程通常有以下四步：

1. 创建拼写纠错干预词典。用户进入到查询分析干预词典页后，点击页面右上角的“创建词典”。选择了词典类型后，为词典命名，干预词典创建完成，词典会出现在页面的词典列表中。
2. 新增和管理干预词典内的干预词条。词典创建完成后，在列表中点击词典名称或点击词典对应的“管理”，即可进入到干预词典的详情页。用户可在详情页内进行干预词条的新增和管理。用户可对Query进行两种类型的干预，-添加纠正词：对一个Query添加纠正词，系统在查询Query时会改写成添加的纠正词进行结果召回。-屏蔽纠正词：对一个Query屏蔽纠正词，系统在查询Query时不再会改写成屏蔽的纠正词进行结果召回。
3. 使用干预词典。创建并填充完成拼写纠错词干预词典后，可在任意应用的查询规则内选择使用。
4. 干预词典效果测试和上线。查询分析规则使用了干预词典后，在应用到线上前建议先进行搜索效果测试，评估效果是否符合干预预期。

实战演示

- 业务场景：某电商导购类业务在OpenSearch的应用实例中配置使用了查询分析规则，规则包含拼写纠错功能，但是在线上发现了badcase，于是决定使用干预功能。
- **Badcase：**用户搜索Query“海澜之谜”，返回的结果不多，但是都完整包含了“海澜之谜”关键词，实际上该查询的相关结果有很多都没有召回。
- 问题诊断：用户的输入Query错误，正确的写法应该是“海蓝之谜”，系统的拼写纠错没有识别出错误写法。
- 解决方案：新建拼写纠错干预词典，在Query“海澜之谜”下干预添加纠正词“海蓝之谜”，再将该拼写纠错词典应用在线上使用的查询分析规则中。
- **配置流程**：

1.点击控制台首页干预功能：查询分析干预词典。

2.创建一个拼写纠错干预词典，命名为"pinxietest"。

3.在"pinxietest"里新增干预词条，Query栏填写“海澜之谜”，纠正词栏填写“海蓝之谜”，干预类型选择“添加”。

4.在app中把干预词典先应用在一个未上线的查询分析规则中，以便进行搜索效果测试。

5.搜索测试是否符合预期效果。发现搜索结果召回了改写成“海蓝之谜”的结果，符合预期。

注意事项

1. 词典类型和名称在创建后均不可修改。
2. 新增干预词条时，填写的Query不应与干预列表内已干预过的Query重复。
3. 新增或修改干预词条后，生效状态如果持续是“正在生效”，可以点击刷新按钮获取生效状态的同步。
4. 同一干预词典可以被多个查询分析规则使用。
5. 由于目前的干预词典是基于系统内置词典的补丁式干预，所以词典使用时会默认勾选使用系统内置词典。
6. 被任一查询规则（不论是否上线）使用的干预词典不能被删除，想要删除需要首先解除使用。

功能限制

1. 拼写纠错干预词典一共可以创建10个。
2. 每次新增拼写纠错干预词条时，Query只支持填写一个，同一Query下纠正词只支持填写一个。
3. 每个拼写纠错干预词典最多可创建500个干预词条。
4. 拼写纠错干预词条按照干预的Query与查询Query精准匹配才生效。例如，为Query“武汗”添加了拼写纠错词“武汉”，那么查询“武汗”时会改写为“武汉”进行召回，但是查询“去武汗”时，不会被改写，仍然按照原Query进行召回。
5. 添加的干预内容均会进行大小写和全半角归一化处理，其中大写字母会归一化为小写，全角会归一化为半角。

搜索测试

通过网页搜索文档

当您的文档正常上传之后，应用管理中搜索按钮也变为可点击状态了。这时您可以进入搜索页面，并且进行搜索。其中搜索语法请参考API开发者手册搜索接口及搜索子句介绍。

通过API/SDK搜索

请参考 开发指南->V3(标准/高级)API参考手册->API概览 及 Java SDK文档及Php SDK文档。

下载中心

开放搜索控制台（下载中心）

SDK下载

提供php、java，具体使用请参考对应的sdk文档。（注意：.net的sdk 目前已下线，不再提供维护。）

搜索结果样式下载

提供几个内置模板的渲染样例，并提供在线效果查看入口。

访问控制 RAM

授权资源类型

资源类型与描述

资源类型	资源描述
apps	acs:opensearch:$regionId:$accountId:$appName
user-analyzers	acs:opensearch:$regionId:$accountId:user-analyzers/$customTokenizerId

$regionId： 表示某个region的id，或使用*代替。

$appName： 表示某个应用名称，或使用*代替。

$customTokenizerId： 表示某个自定义分词器id，或使用*代替。

$accountId： 表示阿里云主账号的数字id，或使用*代替。

支持区域

- 华东1（regionId:cn-hangzhou）
- 华东2（regionId:cn-shanghai）
- 华北1（regionId:cn-qingdao）
- 华北2（regionId:cn-beijing）
- 华南1（regionId:cn-shenzhen）

注意：

- **RAM子账号功能只支持V3及以上SDK版本，V2版SDK不支持RAM子账号功能。**
- 如果期望通过RAM子账号在控制台中配置RDS数据源，必须要先赋予该子账号访问RDS数据源相关权限，否则会报错提示**连接RDS服务失败，请稍后再试**，参考 访问鉴权规则 文档。

访问鉴权规则

您通过云账号创建的OpenSearch应用，都是该账号自己拥有的资源。默认情况下，账号对自己的资源拥有完整的操作权限。

使用阿里云的RAM（Resource Access Management）服务，可以将您云账号下OpenSearch资源的访问及管理权限授予RAM中子用户。

注意：

RAM 子账号功能只支持V3 及以上API（SDK）版本，V2 版API（SDK）不支持 RAM子账号功能。

第三方数据源产品要严格遵守 RAM 权限体系，需要在第三方产品赋予子帐号对应权限。ODPS 数据源不支持 RAM 鉴权，用户需自行与 ODPS 团队沟通完成子账户授权。

使用RAM子账号在控制台中配置 RDS 数据源，必须要再对该RAM子账号进行数据源相关权限授权，否者会报错提示**连接RDS服务失败，请稍后再试**，参考下面的**RDS访问授权**。

以:Search开头的 ACTION 暂不支持 IP条件鉴权，在配置后会有问题，需注意（主要是:SearchApp和:SearchSuggest）。

生效时间

对子用户设置或更新权限配置后，延迟5分钟后生效。

最小常见组合权限

使用RAM子账号登录访问开放搜索控制台，最小常见组合权限包括**应用列表权限**、**应用详情权限**、**监控与报警权限**、**RDS访问授权**等，仅供参考。

应用列表权限

子账号登录后，需要查看控制台应用列表权限。

```
{
"Statement": [
{
"Action": [
"opensearch:ListApp"
],
"Effect": "Allow",
```

```
"Resource": [
"acs:opensearch:*:*:apps/*"
]
}
],
"Version": "1"
}
```

应用详情权限

监控与报警功能是集成在应用详情界面中，因此需要查看应用详情权限，以下示例是查看应用名为**app_schema_demo**的应用详情。

```
{
"Statement": [
{
"Action": [
"opensearch:DescribeApp"
],
"Effect": "Allow",
"Resource": [
"acs:opensearch:*:*:apps/app_schema_demo"
]
}
],
"Version": "1"
}
```

监控与报警权限

监控与报警功能是基于阿里云监控系统，可以通过在RAM策略模板中搜索**AliyunCloudMonitorReadOnlyAccess**来查看云监控只读策略。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"cms:Get*",
"cms:List*",
"cms:Query*",
"cms:BatchQuery*"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"opensearch:ListApps"
],
"Resource": "*",
```

```
"Effect": "Allow"
}
]
}
```

RDS访问授权

访问RDS有两个接口，tables和fields。由于访问RDS需要添加白名单，因此还需要再为RAM子账号设置白名单权限（**如果没有该权限，连接RDS时会报错提示设置RDS的IP白名单失败**）。

RDS 的授权直接在 RAM控制台 配置，可以在概览页配置自定义授权策略或者角色，然后在用户管理页面对子账号进行授权（RDS授权参考文档）。

OpenSearch 使用 RDS 授权最小集合：

- Resource 中的变量含义（例如： $regionid，$accountid，$dbinstanceid 等）。
- Resource 中的内容也可以使用通配符 * 来表示。

```
  {
"Version": "1",
"Statement": [
{
"Action": "rds:DescribeDBInstanceAttribute",
"Resource": "acs:rds:$regionid:$accountid:dbinstance/$dbinstanceid",
"Effect": "Allow"
},
{
"Action": "rds:ModifySecurityIps",
"Resource": "acs:rds:$regionid:$accountid:dbinstance/$dbinstanceid",
"Effect": "Allow"
}
]
}
```

授权参考

在确定要为子用户赋予某些需要操作的应用后，子用户正常登录控制台通常需要依赖多种 action 权限组合，可以考虑赋予子用户 Describe*、List* 权限，当然也可以根据您的实际场景需求为子用户赋予特定的权限组合。

参考（1）

给accountId为1234的主账号下的某个子账号赋予所有区域、所有应用的所有操作权限，该策略在主账号控制台中创建后，需再通过主账号在 RAM 控制台中对子账号授权，或通过 RAM SDK对子账号授权。

1、创建一个策略。

```
{
"Statement": [
```

```
{
"Action": [
"opensearch:*"
],
"Effect": "Allow",
"Resource": [
"acs:opensearch:*:1234:apps/*"
]
}
],
"Version": "1"
}
```

2、把当前策略授权给您指定的子账号。

参考（2）

给accountId为1234的主账号下的某个子账号赋予华东1区域（cn-hangzhou）、所有应用的所有操作权限，该策略在主账号控制台中创建后，需再通过主账号在 RAM 控制台中对子账号授权，或通过 RAM SDK对子账号授权。

1、创建一个策略。

```
{
"Statement": [
{
"Action": [
"opensearch:*"
],
"Effect": "Allow",
"Resource": [
"acs:opensearch:cn-hangzhou:1234:apps/*"
]
}
],
"Version": "1"
}
```

2、把当前策略授权给您指定的子账号。

注意：

在**resource**格式中，如果是通过指定*通配符匹配，将包含所有资源类型。

如果在**resource**格式中，是通过指定应用名匹配， 即使在该策略的Action中指定opensearch:*，也只会包含资源类别为应用名的所有Action，不包含 opensearch:ListApp和opensearch:CreateApp。

每一行**Action**都必须对应所在行的**resource**格式，例如 opensearch:ListApp和opensearch:CreateApp作用范围是所有应用，必须用 *表示。注意这2个**Action**对应**resource**格式和其它**Action**对应 **resource**格式有区别。

如果您的授权策略中只包含指定应用名资源格式，并且您也依赖 opensearch:ListApp和opensearch:CreateApp权限。您需要再创建1个包含这2个资源格式为*的Action策略，并累加授权给指定RAM子账号。

apps授权列表

Action	Action Descripe	resource
opensearch:ListApp	app列表权限	acs:opensearch:$regionId:$accountId:apps/*
opensearch:CreateApp	创建app权限,不限制app name	acs:opensearch:$regionId:$accountId:apps/*
opensearch:DescribeApp	app详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DeleteApp	删除app权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:UpdateApp	app更新权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:SetCurrent	多版本应用切换当前版本服务app	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ReindexApp	app索引重建权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:PushDoc	app推送文档权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:SearchApp	app 查询权限,SearchApp Action鉴权暂不支持ip条件鉴权	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeFirstRank	粗排详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteFirstRank	粗排创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListFirstRank	粗排列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeSecondRank	精排详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteSecondRank	精排创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListSecondRank	精排列表权限	acs:opensearch:$regionId:$accountId:apps/$appName

opensearch:WriteDataSource	数据源创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListDataSource	数据源列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeDataSource	数据源详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteSummary	摘要创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListSummary	摘要列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeSuggest	下拉提示详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteSuggest	下拉提示创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:SearchSuggest	下拉提示搜索权限 ,SearchSuggest Action鉴权暂不支持ip条件鉴权	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ReindexSuggest	下拉提示索引重建权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListSuggest	下拉提示列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteQueryProcessor	qp创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListQueryProcessor	qp 列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeQueryProcessor	qp 详情页权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeTask	任务详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteTask	任务创建，修改，删除权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListTask	任务列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListLog	日志列表权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeQuota	quota详情权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteQuota	quota扩容权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeIndex	全量导入进度权限	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:DescribeOptimizerSlowQuery	查看慢query开通状态	acs:opensearch:$regionId:$accountId:apps/$appName

opensearch:WriteOptimizerSlowQuery	开通（关闭）服务慢query服务	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListOptimizerSlowQueryCategories	列出慢query服务优化建议	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:WriteOptimizerSlowQueryCategories	立即执行慢query分析	acs:opensearch:$regionId:$accountId:apps/$appName
opensearch:ListOptimizerSlowQueries	列出慢query服务中优化建议的Query	acs:opensearch:$regionId:$accountId:apps/$appName

user-analyzers授权列表

Action	Action Descripe	resource
opensearch:*	app列表权限	acs:opensearch:$regionId:$accountId:user-analyzers/*